EXAMPLES IN VERIFICATION OF THE FORTRAN VERIFIER

by

Richard Kwok and Lawrence Yeung

Memorandum No. ERL-M400

August 1973

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

CONTENTS:

(1) SPECIAL ASSERTIONS FOR SNOBOL SIMULATION ROUTINES

1.1) (I..J)

(I..J) is the set (I,I+1,I+2, ... ,J)

1.2) ADDR(I,L)

This function indicates the address(index of SPACE array) of the I-th element of list L.

ELT(I,L) = SPACE(ADDR(I,L)).AND.(1.LE.I,I.LE.LENGTH(L))

1.3) ALL(F(X),X,I,J)

This function implies that F(X) is true for all X in the set (I..J).

ALL(F(X),X,I,J) .IM. (F(X),X.IN.(I..J))

1.4) CAR(L)

CAR(L) is the first element of list L.

CAR(L) = ELT(1,L)

1.5) CDR(L)

CDR(L) is the list of all elements in list L with the first one removed.

CDR(L) = ALL(ELT(X,L),X,2,LENGTH(L))

1.6) CDRN(I,L)

CDRN(0,L) = L .AND. CDRN(I,L) = CDR(CDRN(I-1,L))

1.7) COMP(L2,L3)

This function is the composition of List L3 to list L2.

```
L = COMP(L2,L3).IM. (ALL(ELT(K,L)=ELT(K,L2),K,1,
LENGTH(L2)).OR.ALL(ELT(K,L)=ELT(K-LENGTH(L2),L3),
K,LENGTH(L2)+1,LENGTH(L)),LENGTH(L)=LENGTH(L2)+
LENGTH(L3))
```

1.8) ELT(I,L)

This is the I-th element of list L.

1.9) ERROR(subprogram name)

This function indicates error condition.

1.10) EXISTS(F(X),X,I,J)

This function indicates that there exists an integer X in the range between I and J such that F(X) is true.

```
EXISTS(F(X),X,I,J) .IM. (F(X),X.LE.J,I.LE.X)
```

1.11) EXP(N,I)

This is the exponential function with

```
EXP(N,I) = N**I
```

1.12) LENGTH(L)

This function is the length of list L.

```
LENGTH(L) = (if L=NIL then 0 else LENGTH(CDR(L)+1))
```

1.13) LIST.PIR(PTR,L)

PTR is list pointer in the range defined by list L.

```
LIST.PIR(PTR,L) .IM. EXISTS(K=ADDR(I,L),I,1,LENGTH(L))
```

1.14) LV(LP,L)

This function defines the list value of list pointer LP is list L.

```
LV(LP,L) = (EF(K,(1..N)),((1.LE.K(I),K(I).LE.5000;
SPACE(2*K(I)-1)=L(I),SPACE(2*K(I))=2*K(I+1)-1),
I.IN.(1..N)),LP=2*K(I)-1)
```

1.15) TYPE(L,NO)

This function indicates different type primitives of list L with different values of NO.

```
TYPE(L,NO) = real ,    NO=1
             integer,  NO=2
             string,   NO=3
```

1.16) SUBLIST(J,K,L)

This function indicates sublist of list L with SPACE(J) as first element and element before SPACE(K) as last element.

```
SUBLIST(J,K,L)=SL .ST. (if J=K then SL=NIL else
(CAR(SL)=SPACE(J).AND.CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
```

(1.17) PROPERTIES OF ASSERTIONS

1.17.1) Certain ambiguity of the defined functions are consideration as .TRUE. For example

ALL(F(X),X,I,J).AND.(X.IN.(.NOT.(I..J))) = .TRUE.

EXISTS(F(X),X,I,J).AND.(.NOT.(I.LE.X,X.LE.J)=.TRUE.

1.17.2) length(SUBLIST(I,SPACE(J±1),L))=LENGTH(SUBLIST(I,J,L))+1, for LIST.PIR(J,L).AND.LIST.PIR(I,L).AND.(SPACE(J+1).NE.0)

1.17.3) ALL(F(X),X,I,J).AND.F(J+1) .IM. ALL(F(X),X,I,J+1)
and
EXISTS(F(X),X,I,J).AND.F(J+1) .IM. EXISTS(F(X),X,I,J+1)

1.17.4) To define SUBLIST FUNCTION, LIST.PIR function is needed most of the time.

PROGRAM LISTING AND VERIFICATION EXAMPLES

```
17. 3.13  08/21/7  CUTOUT
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC    .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    ELT(I,L)=CAR(CDR(I-1,L)).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    ALL(F(X),X,I,J) .IN. (F(X), X.IN.(I..J))
CC    EXISTS(F(X), X,I,J) .IN. (F(X), I.LE.X,X.LE.J)
CC    LIST.PIR(P,L)=EXISTS(P=ADDR(K,L),K,1,LENGTH(L)).
CCNO   ((ANC=0 .OR. ANC=1), LV(SLOCAN,L1), LV(PLOCAN,L2)
      SUBROUTINE ANY(SLOCAN,PLOCAN,REGAN,ANC)
      COMMON/BLOCK/SPACE(10000),FAIL,IFOO
      INTEGER SLOCAN,PLOCAN,REGAN,ANC,FAIL,OLDSP,SP,PP,SPACE
      OLDSP = SLOCAN
      SP = SLOCAN
      REGAN=SLOCAN
      IF(SLOCAN) 1,7,1
    1 PP = PLOCAN
CCNO   ((ANC=0).OR.((ANC=1).AND.(SP=SLOCAN))),LIST.PIR(SP,L1),
CCNO   LIST.PIR(PP,L2), ALL(ALL(ELT(I,L1).NE.ELT(J,L2),J,1,LENGTH(L2)),
CCNO   I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))),ALL((SPACE(SP) .NE. ELT(J,L2)),
CCNO   J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))),LV(PLOCAN,L2)
    2 IF(SPACE(SP)-SPACE(PP)) 3,6,3
    3 PP=SPACE(PP+1)
      IF(PP) 2,4,2
    4 IF(ANC) 5,5,7
    5 OLDSP = SP
      SP = SPACE(SP+1)
      IF(SP) 1,7,1
    6 FAIL = 0
      REGAN = SP
CCNO   FAIL=0.EXISTS((SPACE(REGAN)=ELT(K,L2)),K,1,LENGTH(L2)),
CCNO   ALL(ALL(ELT(I,L1) .NE. ELT(J,L2),J,1,LENGTH(L2)),I,1,
CCNO   LENGTH(SUBLIST(SLOCAN,REGAN,L1))),((ANC=0) .OR.
CCNO   ((ANC=1),(REGAN=SLOCAN)))
      RETURN
    7 FAIL = 1
      REGAN = OLDSP
CCNO   FAIL=1, ((ANC=0), ALL(ALL(ELT(I,L1).NE.ELT(J,L2), J,1,LENGTH(L2)),
CCNO   I,1,LENGTH(L1))) .OR.(SLOCAN=0) .OR.
CCNO   (ANC=1, ALL((ELT(I,L1) .NE. ELT(J,L2)), J, 1, LENGTH(L2))))
      RETURN
      END
-----------------------
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND (ANC .EQ. 0 .OR. ANC .EQ. 1) .AND. LV(SLOCAN,L1) .AND.		
COND LV(PLOCAN,L2)		
SUBSCRIPTED ANY(SLOCAN,PLOCAN,BEGAN,ANC)		
COMMON/BLOCK/SPACE(100),FAIL,IFOO		
INTEGER SLOCAN,PLOCAN,BEGAN,ANC,FAIL,OLDSP,SP,PP,SPACE		
OLDSP=SLOCAN	[OLDSP]	[SLOCAN]
SP=SLOCAN	[SP]	[SLOCAN]
BEGAN=SLOCAN	[BEGAN]	[SLOCAN]
IF(SLOCAN .EQ. 0)GO TO 7		
1 PP=PLOCAN	[PP]	[PLOCAN]
COND ((ANC .EQ. 0) .OR. ((ANC .EQ. 1) .AND. (SP .EQ. SLOCAN))) .AND.		
COND LIST.EQ.(SUBLIST(SLOCAN,L1) .AND. LIST.EQ.(SUBLIST(PP,L2) .AND. ALL(ALL(FLT(I,L1) .NE.		
COND FLT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) .AND.		
COND ALL((SPACE(SP) .NE. FLT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2)))		
COND .AND. LV(PLOCAN,L2)		
2 IF(SPACE(SP)-SPACE(PP) .EQ. 0)GO TO 6		
3 PP=SPACE(PP+1)	[PP]	[SPACE,PP]
IF(PP .NE. 0)GO TO 2		
4 IF(ANC .GT. 0)GO TO 7		
5 OLDSP=SP	[OLDSP]	[SP]
SP=SPACE(SP+1)	[SP]	[SPACE,SP]
IF(SP .NE. 0)GO TO 1		
GO TO 7		
6 FAIL=0	[FAIL]	[]
BEGAN=SP	[BEGAN]	[SP]
COND FAIL=0 .AND. EXISTS((SPACE(BEGAN) .EQ. FLT(K,L2)),K,1,LENGTH(L2))		
COND .AND. ALL(ALL(FLT(I,L1) .NE.		
COND FLT(J,L2)),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,BEGAN,L1)))		
COND .AND. ((ANC .EQ. 0) .OR. ((ANC .EQ. 1),(BEGAN .EQ. SLOCAN)))		
RETURN		
7 FAIL=1	[FAIL]	[]
BEGAN=OLDSP	[BEGAN]	[OLDSP]
COND FAIL=1 .AND. ((ANC .EQ. 0,ALL(ALL(FLT(I,L1) .NE.		
COND FLT(I,L2)),J,1,LENGTH(L2)),I,1,LENGTH(L1))) .OR. (SLOCAN .EQ. 0)		
COND .OR. (ANC .EQ. 1,ALL((FLT(I,L1) .NE. FLT(J,L2)),J,1,LENGTH(L2))))		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1		
	COND ((ANC .EQ. 1) .EQ. ANC .EQ. 1) .AND. LV(SLOCAN,L1) .AND. LV(PLOCAN,L2)	
	SUBROUTINE APY(SLOCAN,PLOCAN,BEGAN,ANC)	
	COMMON/BLOCK/SPACE(10000),FAIL,IEQQ	
	INTEGER SLOCAN,PLOCAN,BEGAN,ANC,FAIL,OLDSP,SP,PP,SPACE	
	OLDSP=SLOCAN	
	SP=SLOCAN	
	BEGAN=SLOCAN	
	(SLOCAN .EQ. 0)	
7	. FAIL=1	
	BEGAN=OLDSP	
	COND FAIL=1 .AND. ((ANC .EQ. 0.ALL(ALL(SLT(I,L1) .NE.	
	COND SLT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))) .OR. (SLOCAN .EQ. 0) .OR. (ANC	
	COND .EQ. 1.ALL((SLT(I,L1) .NE. SLT(J,L2)),J,1,LENGTH(L2))))	
		(((0 .EQ. ANC) .OR. (0 .EQ. 1-ANC)) .AND. (LV(SLOCAN,L1)) .AND. (LV(PLOCAN,L2)) .AND. (0 .EQ. SLOCAN)) .IMPLIES. (((0 .EQ. -ANC+0.ALL(ALL(0 .NE. -SLT(I,L1)+SLT(J,L2),J,1,LENGTH(L2)), I,1,LENGTH(L1))) .OR. (0 .EQ. SLOCAN)) .OR. (0 .EQ. -ANC+1,ALL(0 .NE. -SLT(1,L1)+SLT(J,L2),J,1,LENGTH(L2))))
2		
	COND ((ANC .EQ. 0 .OR. ANC .EQ. 1) .AND. LV(SLOCAN,L1) .AND. LV(PLOCAN,L2)	
	SUBROUTINE APY(SLOCAN,PLOCAN,BEGAN,ANC)	
	COMMON/BLOCK/SPACE(10000),FAIL,IEQQ	
	INTEGER SLOCAN,PLOCAN,BEGAN,ANC,FAIL,OLDSP,SP,PP,SPACE	
	OLDSP=SLOCAN	
	SP=SLOCAN	
	BEGAN=SLOCAN	
	.NOT. (SLOCAN .EQ. 0)	
1	PP=PLOCAN	
	COND ((ANC .EQ. 0) .OR. ((ANC .EQ. 1) .AND. (SP .EQ. SLOCAN))) .AND.	
	COND LIST.PIR(SP,L1) .AND. LIST.PIR(PP,L2) .AND. ALL(ALL(ELT(I,L1) .NE.	
	COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) .AND.	
	COND ALL((SPACE(SP) .EQ. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) .AND.	
	COND LV(PLOCAN,L2)	
		(((0 .EQ. ANC) .OR. (0 .EQ. 1-ANC)) .AND. (LV(SLOCAN,L1)) .AND. (LV(PLOCAN,L2)) .AND. (0 .NE. SLOCAN)) .IMPLIES. ((LIST.PIR(SLOCAN,L1)) .AND. (LIST.PIR(PLOCAN,L2)) .AND. (LV(PLOCAN,L2)) .AND. (ALL(0 .NE.

```
COND ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) .AND.
COND LV(PLOCAN,L2)
                                                                  (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                  .AND. (0 .EQ. SLOCAN-SP))) .AND.
                                                                  (LIST.PIR(SP,L1)) .AND.
                                                                  (LIST.PIR(PP,L2)) .AND. (ALL(ALL(0
                                                                  .NE.
                                                                  -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                  I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))))
                                                                  .AND. (ALL(0 .NE.
                                                                  ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                                  T(PLOCAN,PP,L2)))) .AND.
                                                                  (LV(PLOCAN,L2)) .AND. (0 .NE.
                                                                  -SPACE(PP)+SPACE(SP)) .AND. (0 .NE.
                                                                  -SPACE(1+PP)))
                                                                   .IMPLIES.
                                                                  (((LIST.PIR(SP,L1)) .AND.
                                                                  (LIST.PIR(SPACE(1+PP),L2)) .AND.
                                                                  (LV(PLOCAN,L2)) .AND. (ALL(0 .NE.
                                                                  ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                                  T(PLOCAN,SPACE(1+PP),L2)))) .AND.
                                                                  (ALL(ALL(0 .NE.
                                                                  -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                  I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))))
                                                                  .AND. ((0 .EQ. ANC) .OR. ((0 .EQ.
                                                                  1-ANC) .AND. (0 .EQ. SLOCAN-SP))))
5
      COND ((ANC .EQ. 0) .OR. ((ANC .EQ. 1) .AND. (SP .EQ. SLOCAN))) .AND.
      COND LIST.PIR(SP,L1) .AND. LIST.PIR(PP,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),I,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) .AND.
      COND ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) .AND.
      COND LV(PLOCAN,L2)
      2       .NOT. (SPACE(SP)-SPACE(PP) .EQ. 0)
      3       PP=SPACE(PP+1)
               .NOT. (PP .NE. 0)
      4       (ANC .GT. 0)
      7       FAIL=1
              SLOCAN=SLOCAN
      COND FAIL=1 .AND. ((ANC .EQ. 0,ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))) .OR. (SLOCAN .EQ. 0) .OR. (ANC
      COND .EQ. 1,ALL((ELT(1,L1) .NE. ELT(J,L2)),J,1,LENGTH(L2))))
                                                                  (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                  .AND. (0 .EQ. SLOCAN-SP))) .AND.
                                                                  (LIST.PIR(SP,L1)) .AND.
                                                                  (LIST.PIR(PP,L2)) .AND. (ALL(ALL(0
                                                                  .NE.
                                                                  -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                  I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))))
                                                                  .AND. (ALL(0 .NE.
                                                                  ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                                  T(PLOCAN,PP,L2)))) .AND.
                                                                  (LV(PLOCAN,L2)) .AND. (0 .NE.
                                                                  -SPACE(PP)+SPACE(SP)) .AND. (0 .EQ.
                                                                  -SPACE(1+PP)) .AND. (0 .LT. ANC))
```

```
                                                                .IMPLIES.                                                        1
                                                                (((0 .EQ. -ANC+0,ALL(ALL(0 .NE.
                                                                -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                I,1,LENGTH(L1))) .OR. (0 .EQ.
                                                                SLOCAN)) .OR. (0 .EQ. -ANC+1,ALL(0
                                                                .NE.
                                                                -ELT(1,L1)+ELT(J,L2),J,1,LENGTH(L2)))                            2
                                                                )

6
      COND ((ANC .EQ. 0) .OR. ((ANC .EQ. 1) .AND. (SP .EQ. SLOCAN))) .AND.
      COND LIST.PIR(SP,L1) .AND. LIST.PIR(PP,L2) .AND. ALL(ALL(ELT(I,L1) .NE.                                                    3
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) .AND.
      COND ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) .AND.
      COND LV(PLOCAN,L2)
      2      .NOT. (SPACE(SP)-SPACE(PP) .EQ. 0)
      3      PP=SPACE(PP+1)
              .NOT. (PP .LE. 0)
      4      .NOT. (ANC .GT. 0)
      5      CLOSP=SP
             SP=SPACE(SP+1)
             (SP .NE. 0)
      1      PP=PLOCAN
      COND ((ANC .EQ. 0) .OR. ((ANC .EQ. 1) .AND. (SP .EQ. SLOCAN))) .AND.
      COND LIST.PIR(SP,L1) .AND. LIST.PIR(PP,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) .AND.
      COND ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) .AND.
      COND LV(PLOCAN,L2)
                                                                (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                .AND. (0 .EQ. SLOCAN-SP))) .AND.
                                                                (LIST.PIR(SP,L1)) .AND.
                                                                (LIST.PIR(PP,L2)) .AND. (ALL(ALL(0
                                                                .NE.
                                                                -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))))
                                                                .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                                T(PLOCAN,PP,L2)))) .AND.
                                                                (LV(PLOCAN,L2)) .AND. (0 .NE.
                                                                -SPACE(PP)+SPACE(SP)) .AND. (0 .EQ.
                                                                -SPACE(1+PP)) .AND. (0 .LE. -ANC)
                                                                .AND. (0 .NE. -SPACE(1+SP)))
                                                                 .IMPLIES.
                                                                ((LIST.PIR(SPACE(1+SP),L1)) .AND.
                                                                (LIST.PIR(PLOCAN,L2)) .AND.
                                                                (LV(PLOCAN,L2)) .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(SPACE(1+SP)),J,1,LENG
                                                                TH(SUBLIST(PLOCAN,PLOCAN,L2))))
                                                                .AND. (ALL(ALL(0 .NE.
                                                                -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                I,1,LENGTH(SUBLIST(SLOCAN,SPACE(1+SP)
                                                                ,L1)))) .AND. ((0 .EQ. ANC) .OR. ((0
                                                                .EQ. 1-ANC) .AND. (0 .EQ.
                                                                SLOCAN-SPACE(1+SP)))))
```

```
7
      COND ((ANC .EQ. 0) .OR. ((ANC .EQ. 1) .AND. (SP .EQ. SLOCAN))) .AND.
      COND LIST.PIR(SP,L1) .AND. LIST.PIR(PP,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) .AND.
      COND ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) .AND.
      COND LV(PLOCAN,L2)
      2       .NOT. (SPACE(SP)-SPACE(PP) .EQ. 0)
      3     PP=SPACE(PP+1)
             .NOT. (PP .NE. 0)
      4      .NOT. (ANC .GT. 0)
      5     OLDSP=SP
            SP=SPACE(SP+1)
             .NOT. (SP .NE. 0)
      7     FAIL=1
            PLOCAN=OLDSP
      COND FAIL=1 .AND. ((ANC .EQ. 0,ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))) .OR. (SLOCAN .EQ. 0) .OR. (ANC
      COND .EQ. 1,ALL((ELT(I,L1) .NE. ELT(J,L2)),J,1,LENGTH(L2))))
```

```
(((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
.AND. (0 .EQ. SLOCAN-SP))) .AND.
(LIST.PIR(SP,L1)) .AND.
(LIST.PIR(PP,L2)) .AND. (ALL(ALL(0
.NE.
-ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))))
.AND. (ALL(0 .NE.
ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
T(PLOCAN,PP,L2)))) .AND.
(LV(PLOCAN,L2)) .AND. (0 .NE.
-SPACE(PP)+SPACE(SP)) .AND. (0 .EQ.
-SPACE(1+PP)) .AND. (0 .LE. -ANC)
.AND. (0 .EQ. -SPACE(1+SP)))
 .IMPLIES.
(((0 .EQ. -ANC+0,ALL(ALL(0 .NE.
-ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
I,1,LENGTH(L1))) .OR. (0 .EQ.
SLOCAN)) .OR. (0 .EQ. -ANC+1,ALL(0
.NE.
-ELT(1,L1)+ELT(J,L2),J,1,LENGTH(L2)))
)
```

2.1) ANY(SLOCAN,PLOCAN,BEGAN,ANC)

This subroutine works like the ANY function of SNOBOL.
SLOCAN=subject string
PLOCAN=pattern string
BEGAN =pointer to sucessfully matched character.
ANC =0 or 1 for unanchored or anchored mode.

VERIFICATION:

Path 1: Since SLOCAN=0, the conclusion holds.

Path 2: Since SUBLIST(PLOCAN,PLOCAN,L2) and SUBLIST(SLOCAN,SLOCAN,L1), SL1=nil and SL2=nil. Hence the assertion holds.

Path 3: Since SPACE(SP)=SPACE(PP) and LIST.PIR(PP,L2), hence EXISTS(SPACE(SP)=ELT(K,L2),K,1,LENGTH(L2)).

Path 4: SINCE ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) and SPACE(SP)≠SPACE(PP) and LIST.PIR(PP,L2), therefore ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(SUBLIST(PLOCAN,SPACE(PP+1),L2))).

Path 5: Since ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(SUBLIST(PLOCAN,PP,L2))) and SPACE(SP)≠SPACE(PP) and SPACE(PP+1)=0, hence ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(L2)). For ANC=1, ALL(ELT(1,L1)≠ELT(J,L2),J,1,LENGTH(L2)) is true.

Path 6: For ANC=0, ALL(ALL(ELT(I,L1)≠ELT(J,L2),J,1,LENGTH(L2)) I,1,LENGTH(SUBLIST(SLOCAN,SP,L1))) AND ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(SUBLIST(PLOCAN, PP,L2))) and SPACE(SP)≠SPACE(PP) and SPACE(PP+1)=0 implies ALL(ALL(ELT(I,L1)≠ELT(J,L2),J,1,LENGTH(L2)), I,1,LENGTH(SUBLIST(SLOCAN,SPACE(SP+1),L1))).

Path 7: Similar to path 6 and with SPACE(SP+1)=0 hence for ANC=0, ALL(ALL(ELT(I,L1)≠ELT(J,L2),J,1,LENGTH(L2)), I,1,LENGTH(L1)).

```
CC     LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC     ANC = 0 .OR. 1
COND    LV(SLOCB,L1)
       SUBROUTINE BAL1(SLOCB,BEGB,ENDB,ANC)
       COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
       INTEGER SLOCB,BEGB,ENDB,ANC,FAIL,SPACE,S,FP,BP,OLDBP
       COMMON/ALPHNU/Q(63)
       INTEGER Q
       IRL=Q(41)
       IRR=Q(42)
       OLDBP = SLOCB
       S=0
       BP=SLOCB
       IF(SLOCB) 1,11,1
     1 BP=BP
2      IF(SPACE(BP)-IRL)3,5,3
COND    SPACE(BP) .NE. Q(41),IRR=Q(42),IRL=Q(41)
3      IF(SPACE(BP)-IRR)4,7,4
COND    SPACE(BP) .NE. Q(41), SPACE(BP) .NE. Q(42),
COND    IRL=Q(41),IRR=Q(42)
     4 IF(S) 6,12,6
COND    SPACE(BP)=Q(41),IRL=Q(41),IRR=Q(42)
     5 S=S+1
     6 OLDBP=BP
       BP = SPACE(BP+1)
       IF(BP) 2,9,2
COND    SPACE(BP)=Q(42),IRL=Q(41),IRR=Q(42)
     7 S=S-1
       IF(S) 8,12,6
     8 OLDBP = BP
       S= 0
     9 IF(ANC) 10,10,11
COND    ANC .LE. 0,IRL=Q(41),IRR=Q(42)
    10 BP = SPACE(BP+1)
       IF(BP) 1,11,1
    11 FAIL=1
       BEGB=SLOCB
       ENDB=OLDBP
COND    FAIL=1
       RETURN
    12 FAIL=0
       BEGB=BP
       ENDB=BP
COND    FAIL=0,S=0
       RETURN
       END
```

	PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
	COND LV(SLOCB,L1)		
	SUBROUTINE BAL1(SLOCB,BEGB,ENDB,ANC)		
	COMMON/BLOCK/SPACE(10000),FAIL,IFQQ		
	INTEGER SLOCB,BEGB,ENDB,ANC,FAIL,SPACE,S,FP,BP,OLDBP		
	COMMON/ALPHNU/Q(63)		
	INTEGER Q		
	IRL=Q(41)	[IRL]	[Q]
	IRR=Q(42)	[IRR]	[Q]
	OLDBP=SLOCB	[OLDBP]	[SLOCB]
	S=0	[S]	[]
	FP=SLOCB	[FP]	[SLOCB]
	IF(SLOCB .EQ. 0)GO TO 11		
1	BP=FP	[BP]	[FP]
2	IF(SPACE(BP)-IRL .EQ. 0)GO TO 5		
COND	SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)		
3	IF(SPACE(BP)-IRR .EQ. 0)GO TO 7		
COND	SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41)		
COND	.AND. IRR=Q(42)		
4	IF(S .NE. 0)GO TO 6		
	GO TO 12		
COND	SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)		
5	S=S+1	[S]	[S]
6	OLDBP=BP	[OLDBP]	[BP]
	BP=SPACE(BP+1)	[BP]	[SPACE,BP]
	IF(BP .NE. 0)GO TO 2		
	GO TO 9		
COND	SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)		
7	S=S-1	[S]	[S]
	IF(S .EQ. 0)GO TO 12		
	IF(S .GT. 0)GO TO 6		
8	OLDBP=BP	[OLDBP]	[BP]
	S=0	[S]	[]
9	IF(ANC .GT. 0)GO TO 11		
COND	ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)		
10	FP=SPACE(FP+1)	[FP]	[SPACE,FP]
	IF(FP .NE. 0)GO TO 1		
11	FAIL=1	[FAIL]	[]
	BEGB=SLOCB	[BEGB]	[SLOCB]
	ENDB=OLDBP	[ENDB]	[OLDBP]
COND	FAIL=1		
	RETURN		
12	FAIL=0	[FAIL]	[]
	BEGB=FP	[BEGB]	[FP]
	ENDB=BP	[ENDB]	[BP]
COND	FAIL=0 .AND. S=0		
	RETURN		
	END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(SLOCB,L1) SUBROUTINE BAL1(SLOCB,BEGB,ENDB,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCB,BEGB,ENDB,ANC,FAIL,SPACE,S,FP,BP,OLDBP COMMON/ALPHNU/Q(63) INTEGER Q IRL=Q(41) IRR=Q(42) OLDBP=SLOCB S=0 FP=SLOCB (SLOCB .EQ. 0) 11 FAIL=1 BEGB=SLOCB ENDB=OLDBP COND FAIL=1	NONE
2	COND LV(SLOCB,L1) SUBROUTINE BAL1(SLOCB,BEGB,ENDB,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCB,BEGB,ENDB,ANC,FAIL,SPACE,S,FP,BP,OLDBP COMMON/ALPHNU/Q(63) INTEGER Q IRL=Q(41) IRR=Q(42) OLDBP=SLOCB S=0 FP=SLOCB .NOT. (SLOCB .EQ. 0) 1 BP=FP 2 (SPACE(BP)-IRL .EQ. 0) COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)	((LV(SLOCB,L1)) .AND. (0 .NE. SLOCB) .AND. (0 .EQ. -Q(41)+SPACE(SLOCB))) .IMPLIES. ((0 .EQ. Q(41)-SPACE(SLOCB)))
3	COND LV(SLOCB,L1) SUBROUTINE BAL1(SLOCB,BEGB,ENDB,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCB,BEGB,ENDB,ANC,FAIL,SPACE,S,FP,BP,OLDBP COMMON/ALPHNU/Q(63) INTEGER Q IRL=Q(41) IRR=Q(42)	

```
        OLDBP=SLOCB
        S=0
        FP=SLOCB
          .NOT. (SLOCB .EQ. 0)
1       BP=FP
2         .NOT. (SPACE(BP)-IRL .EQ. 0)
COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
                                                        ((LV(SLOCB,L1)) .AND. (0 .NE. SLOCB)
                                                        .AND. (0 .NE. -Q(41)+SPACE(SLOCB)))
                                                         .IMPLIES.
                                                        ((0 .NE. Q(41)-SPACE(SLOCB)))

4
COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
3       (SPACE(BP)-IRR .EQ. 0)
COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
                                                        ((0 .NE. Q(41)-SPACE(BP)) .AND. (0
                                                        .EQ. -IRR+Q(42)) .AND. (0 .EQ.
                                                        -IRL+Q(41)) .AND. (0 .EQ.
                                                        -IRR+SPACE(BP)))
                                                         .IMPLIES.
                                                        ((0 .EQ. Q(42)-SPACE(BP)))

5
COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
3       .NOT. (SPACE(BP)-IRR .EQ. 0)
COND SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41) .AND.
COND IRR=Q(42)
                                                        ((0 .NE. Q(41)-SPACE(BP)) .AND. (0
                                                        .EQ. -IRR+Q(42)) .AND. (0 .EQ.
                                                        -IRL+Q(41)) .AND. (0 .NE.
                                                        -IRR+SPACE(BP)))
                                                         .IMPLIES.
                                                        ((0 .NE. Q(42)-SPACE(BP)) .AND. (0
                                                        .NE. Q(41)-SPACE(BP)))

6
COND SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41) .AND.
COND IRR=Q(42)
4       (S .NE. 0)
6       (LOOP=BP
        BP=SPACE(BP+1)
        (BP .NE. 0)
2       (SPACE(BP)-IRL .EQ. 0)
COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
                                                        ((0 .NE. Q(41)-SPACE(BP)) .AND. (0
                                                        .NE. Q(42)-SPACE(BP)) .AND. (0 .EQ.
                                                        -IRL+Q(41)) .AND. (0 .EQ.
                                                        -IRR+Q(42)) .AND. (0 .NE. -S) .AND.
                                                        (0 .NE. -SPACE(1+BP)) .AND. (0 .EQ.
                                                        -IRL+SPACE(SPACE(1+BP))))
                                                         .IMPLIES.
                                                        ((0 .EQ. Q(41)-SPACE(SPACE(1+BP))))
```

```
7
      COND SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41) .AND.
      COND IRR=Q(42)
      4       (S .NE. 0)
      6       CLDBP=BP
              BP=SPACE(BP+1)
              (BP .NE. 0)
      2        .NOT. (SPACE(BP)-IRL .EQ. 0)
      COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
                                                    ((0 .NE. Q(41)-SPACE(BP)) .AND. (0
                                                    .NE. Q(42)-SPACE(BP)) .AND. (0 .EQ.
                                                    -IRL+Q(41)) .AND. (0 .EQ.
                                                    -IRR+Q(42)) .AND. (0 .NE. -S) .AND.
                                                    (0 .NE. -SPACE(1+BP)) .AND. (0 .NE.
                                                    -IRL+SPACE(SPACE(1+BP))))
                                                     .IMPLIES.
                                                    ((0 .NE. Q(41)-SPACE(SPACE(1+BP))))

8
      COND SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41) .AND.
      COND IRR=Q(42)
      4       (S .NE. 0)
      6       CLDBP=BP
              BP=SPACE(BP+1)
               .NOT. (BP .NE. 0)
      9       (ANC .GT. 0)
      11      FAIL=1
              BEGB=SLDCB
              ENDB=CLDBP
      COND FAIL=1
                                                    NONE

9
      COND SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41) .AND.
      COND IRR=Q(42)
      4       (S .NE. 0)
      6       CLDBP=BP
              BP=SPACE(BP+1)
               .NOT. (BP .NE. 0)
      9        .NOT. (ANC .GT. 0)
      COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)
                                                    NONE

10
      COND SPACE(BP) .NE. Q(41) .AND. SPACE(BP) .NE. Q(42) .AND. IRL=Q(41) .AND.
      COND IRR=Q(42)
      4        .NOT. (S .NE. 0)
      12      FAIL=0
              BEGB=FP
              ENDB=BP
      COND FAIL=0 .AND. S=0
```

```
                                                                    ((0 .NE. Q(41)-SPACE(BP)) .AND. (0
                                                                    .NE. Q(42)-SPACE(BP)) .AND. (0 .EQ.
                                                                    -IRL+Q(41)) .AND. (0 .EQ.
                                                                    -IRR+Q(42)) .AND. (0 .EQ. -S))
                                                                     .IMPLIES.
                                                                    ((0 .EQ. S))

11
      COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
      5     S=S+1
      6     OLDBP=BP
            BP=SPACE(BP+1)
            (BP .NE. 0)
      2     (SPACE(BP)-IRL .EQ. 0)
      COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
                                                                    ((0 .EQ. Q(41)-SPACE(BP)) .AND. (0
                                                                    .EQ. -IRL+Q(41)) .AND. (0 .EQ.
                                                                    -IRR+Q(42)) .AND. (0 .NE.
                                                                    -SPACE(1+BP)) .AND. (0 .EQ.
                                                                    -IRL+SPACE(SPACE(1+BP))))
                                                                     .IMPLIES.
                                                                    ((0 .EQ. Q(41)-SPACE(SPACE(1+BP))))

12
      COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
      5     S=S+1
      6     OLDBP=BP
            BP=SPACE(BP+1)
            (BP .NE. 0)
      2      .NOT. (SPACE(BP)-IRL .EQ. 0)
      COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
                                                                    ((0 .EQ. Q(41)-SPACE(BP)) .AND. (0
                                                                    .EQ. -IRL+Q(41)) .AND. (0 .EQ.
                                                                    -IRR+Q(42)) .AND. (0 .NE.
                                                                    -SPACE(1+BP)) .AND. (0 .NE.
                                                                    -IRL+SPACE(SPACE(1+BP))))
                                                                     .IMPLIES.
                                                                    ((0 .NE. Q(41)-SPACE(SPACE(1+BP))))

13
      COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
      5     S=S+1
      6     OLDBP=BP
            BP=SPACE(BP+1)
             .NOT. (BP .NE. 0)
      9     (ANC .GT. 0)
      11    FAIL=1
            BEGB=SLOCB
            ENDB=OLDBP
      COND FAIL=1                                                   NONE
```

```
14
      COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
5     S=S+1
6     OLDBP=BP
      BP=SPACE(BP+1)
       .NOT. (BP .NE. 0)
9      .NOT. (ANC .GT. 0)
      COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)                    NONE

15
      COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
7     S=S-1
      (S .EQ. 0)
12    FAIL=0
      BEGB=BP
      ENDB=BP
      COND FAIL=0 .AND. S=0                                              NONE

16
      COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
7     S=S-1
       .NOT. (S .EQ. 0)
      (S .GT. 0)
6     OLDBP=BP
      BP=SPACE(BP+1)
      (BP .NE. 0)
2     (SPACE(BP)-IRL .EQ. 0)
      COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
                                                                         ((0 .EQ. Q(42)-SPACE(BP)) .AND. (0
                                                                         .EQ. -IRL+Q(41)) .AND. (0 .EQ.
                                                                         -IRR+Q(42)) .AND. (0 .NE. -1+S)
                                                                         .AND. (0 .LT. -1+S) .AND. (0 .NE.
                                                                         -SPACE(1+BP)) .AND. (0 .EQ.
                                                                         -IRL+SPACE(SPACE(1+BP))))
                                                                          .IMPLIES.
                                                                         ((0 .EQ. Q(41)-SPACE(SPACE(1+BP))))

17
3     COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
7     S=S-1
       .NOT. (S .EQ. 0)
      (S .GT. 0)
6     OLDBP=BP
2     BP=SPACE(BP+1)
      (BP .NE. 0)
2      .NOT. (SPACE(BP)-IRL .EQ. 0)
      COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
                                                                         ((0 .EQ. Q(42)-SPACE(BP)) .AND. (0
                                                                         .EQ. -IRL+Q(41)) .AND. (0 .EQ.
                                                                         -IRR+Q(42)) .AND. (0 .NE. -1+S)
1                                                                        .AND. (0 .LT. -1+S) .AND. (0 .NE.
```

```
                                                                  -SPACE(1+BP)) .AND. (0 .NE.
                                                                  -IRL+SPACE(SPACE(1+BP))))
                                                                   .IMPLIES.
                                                                  ((0 .NE. Q(41)-SPACE(SPACE(1+BP))))

18
        COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
        7       S=S-1
                 .NOT. (S .EQ. 0)
                (S .GT. 0)
        6       OLDBP=BP
                BP=SPACE(BP+1)
                 .NOT. (BP .NE. 0)
        9       (ANC .GT. 0)
        11      FAIL=1
                BEGB=SLOCB
                ENDB=OLDBP
        COND FAIL=1                                               NONE

19
        COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
        7       S=S-1
                 .NOT. (S .EQ. 0)
                (S .GT. 0)
        6       OLDBP=BP
                BP=SPACE(BP+1)
                 .NOT. (BP .NE. 0)
        9        .NOT. (ANC .GT. 0)
        COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)           NONE

20
        COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
        7       S=S-1
                 .NOT. (S .EQ. 0)
                 .NOT. (S .GT. 0)
        8       OLDBP=BP
                S=0
        9       (ANC .GT. 0)
        11      FAIL=1
                BEGB=SLOCB
                ENDB=OLDBP
        COND FAIL=1                                               NONE

21
        COND SPACE(BP)=Q(42) .AND. IRL=Q(41) .AND. IRR=Q(42)
        7       S=S-1
                 .NOT. (S .EQ. 0)
                 .NOT. (S .GT. 0)
        8       OLDBP=BP
```

```
      S=0
9       .NOT. (ANC .GT. 0)
COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)                    NONE
```

```
22
      COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)
10    FP=SPACE(FP+1)
      (FP .NE. 0)
1     BP=FP
2     (SPACE(BP)-IRL .EQ. 0)
COND SPACE(BP)=Q(41) .AND. IRL=Q(41) .AND. IRR=Q(42)
                                                                   ((0 .LE. -ANC) .AND. (0 .EQ.
                                                                   -IRL+Q(41)) .AND. (0 .EQ.
                                                                   -IRR+Q(42)) .AND. (0 .NE.
                                                                   -SPACE(1+FP)) .AND. (0 .EQ.
                                                                   -IRL+SPACE(SPACE(1+FP))))
                                                                    .IMPLIES.
                                                                   ((0 .EQ. Q(41)-SPACE(SPACE(1+FP))))
```

```
23
      COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)
10    FP=SPACE(FP+1)
      (FP .NE. 0)
1     BP=FP
2       .NOT. (SPACE(BP)-IRL .EQ. 0)
COND SPACE(BP) .NE. Q(41) .AND. IRR=Q(42) .AND. IRL=Q(41)
                                                                   ((0 .LE. -ANC) .AND. (0 .EQ.
                                                                   -IRL+Q(41)) .AND. (0 .EQ.
                                                                   -IRR+Q(42)) .AND. (0 .NE.
                                                                   -SPACE(1+FP)) .AND. (0 .NE.
                                                                   -IRL+SPACE(SPACE(1+FP))))
                                                                    .IMPLIES.
                                                                   ((0 .NE. Q(41)-SPACE(SPACE(1+FP))))
```

```
24
      COND ANC .LE. 0 .AND. IRL=Q(41) .AND. IRR=Q(42)
10    FP=SPACE(FP+1)
        .NOT. (FP .NE. 0)
11    FAIL=1
      BEGB=SLOCB
      ENDB=OLDBP
COND FAIL=1                                                        NONE
```

2.2) BAL1(SIOCB,BEGB,ENDB,ANC)

This subroutine finds the first balanced expression in the subject string.
SIOCB = subject string
BEGB = pointer to the first character of the balanced expression.
ENDB = pointer to the last character of the balanced expression found.

VERIFICATION:

Path 1: proved.

Path 2: obvious.

Path 3: obvious.

Path 4: Since SPACE(BP)=IRR and IRR=Q(42), hence SPACE(BP)=Q(42).

Path 5: can be verified using the same substution in the above path.

Path 6: Since SPACE(SPACE(BP+1))=IRL and IRL=Q(41), therefore SPACE(SPACE(BP+1))=Q(41).

Path 7: same as path 6.

Path 8: proved.

Path 9: proved.

Path 10: obvious.

Path 11: same as path 6.

Path 12: same as path 11.

Path 13:proved

Path 14: proved.

Path 15: proved.

Path 16: same as path 6.

Path 17: same as path 16.

Path 18: proved.

Path 19: proved.

Path 20: proved.

Path 21:proved.

Path 22: same as path 6.

Path 23: same as path 22

Path 24: proved.

```
CC  NUMB(X,L)=SUM((IF ELT(I,L)=X THEN 1 ELSE 0),I,1,LENGTH(L))
CC  DIFLR(BP1,X,L)= NUMB(Q(41),SUBLIST(BP1,X,L)) - NUMB(Q(42),
CC  SUBLIST(BP1,X,L))
C
COND  LV(BP1,L), LIST.PTR(BP2,L)
      SUBROUTINE BAL2(BP1,BP2)
      COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
      INTEGER BP1,BP2,S,FP,SPACE,FAIL
      COMMON/ALPHNL/Q(63)
      INTEGER Q
      IRL=Q(41)
      IRR=Q(42)
      S=0
      FP=BP1
      IF(BP1-BP2) 1,8,1
COND  (LIST.PTR(BP2,L),LV(BP1,L),(BP1.NE.BP2)),IRL=Q(41),IRR=Q(42),
COND  LIST.PTR(FP,SUBLIST(BP1,BP2,L)),((S=DIFLR(BP1,FP,L)),(S.GE.0))
1     IF(SPACE(FP)-IRL)2,3,2
2     IF(SPACE(FP)-IRR)4,6,4
    3 S=S+1
    4 FP = SPACE(FP+1)
      IF(FP-BP2) 5,7,5
    5 IF(FP) 1,9,1
    6 S=S-1
      IF(S) 8,4,4
    7 IF(S) 8,9,8
    8 FAIL=1
COND  FAIL=1,LIST.PTR(BP2,L),((BP1=BP2).OR.((DIFLR(BP1,BP2,L).NE.0).OR.
COND  ((DIFLR(BP1,FP,L).LT.0),LIST.PTR(FP,SUBLIST(BP1,BP2,L))),
COND  (BP1.NE.BP2)))
      RETURN
    9 FAIL = 0
COND  FAIL=0,LIST.PTR(BP2,L),((DIFLR(BP1,BP2,L)=0),(BP1.NE.BP2))
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(BP1,L) .AND. LIST.PTR(BP2,L)		
SUBROUTINE BAL2(BP1,BP2)		
COMMON/BLOCK/SPACE(10000),FAIL,IFQQ		
INTEGER BP1,BP2,S,FP,SPACE,FAIL		
COMMON/ALPHNU/Q(63)		
INTEGER Q		
IRL=Q(41)	[IRL]	[Q]
IRR=Q(42)	[IRR]	[Q]
S=0	[S]	[ ]
FP=BP1	[FP]	[BP1]
IF(BP1-BP2 .EQ. 0)GO TO 8		
COND (LIST.PTR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND.		
COND IRR=Q(42) .AND. LIST.PTR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ.		
COND DIFLR(BP1,FP,L)),(S .GE. 0))		
1 IF(SPACE(FP)-IRL .EQ. 0)GO TO 3		
2 IF(SPACE(FP)-IRR .NE. 0)GO TO 4		
GO TO 6		
3 S=S+1	[S]	[S]
4 FP=SPACE(FP+1)	[FP]	[SPACE,FP]
IF(FP-BP2 .EQ. 0)GO TO 7		
5 IF(FP .NE. 0)GO TO 1		
GO TO 9		
6 S=S-1	[S]	[S]
IF(S .LT. 0)GO TO 8		
GO TO 4		
7 IF(S .EQ. 0)GO TO 9		
8 FAIL=1	[FAIL]	[ ]
COND FAIL=1 .AND. LIST.PTR(BP2,L) .AND. ((BP1 .EQ. BP2) .OR.		
COND ((DIFLR(BP1,BP2,L) .NE. 0) .OR. ((DIFLR(BP1,FP,L) .LT.		
COND 0),LIST.PTR(FP,SUBLIST(BP1,BP2,L))),(BP1 .NE. BP2)))		
RETURN		
9 FAIL=0	[FAIL]	[ ]
COND FAIL=0 .AND. LIST.PTR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1		
COND .NE. BP2))		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(BP1,L) .AND. LIST.PIR(BP2,L) SUBROUTINE BAL2(BP1,BP2) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER BP1,BP2,S,FP,SPACE,FAIL COMMON/ALPHNU/Q(63) INTEGER Q IRL=Q(41) IRR=Q(42) S=0 FP=BP1 (BP1-BP2 .EQ. 0) ' 8 FAIL=1 COND FAIL=1 .AND. LIST.PIR(BP2,L) .AND. ((BP1 .EQ. BP2) .OR. ((DIFLR(BP1,BP2,L) COND .NE. 0) .OR. ((DIFLR(BP1,FP,L) .LT. COND 0),LIST.PIR(FP,SUBLIST(BP1,BP2,L))),(BP1 .NE. BP2)))	((LV(BP1,L)) .AND. (LIST.PIR(BP2,L)) .AND. (0 .EQ. BP1-BP2)) .IMPLIES. ((LIST.PIR(BP2,L)) .AND. ((0 .EQ. -BP1+BP2) .OR. ((0 .NE. -DIFLR(BP1,BP2,L)) .OR. (0 .LT. -DIFLR(BP1,BP1,L),LIST.PIR(BP1,SUBLIS T(BP1,BP2,L)),0 .NE. -BP1+BP2))))
2	COND LV(BP1,L) .AND. LIST.PIR(BP2,L) SUBROUTINE BAL2(BP1,BP2) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER BP1,BP2,S,FP,SPACE,FAIL COMMON/ALPHNU/Q(63) INTEGER Q IRL=Q(41) IRR=Q(42) S=0 FP=BP1 .NOT. (BP1-BP2 .EQ. 0) COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42) COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S COND .GE. 0))	((LV(BP1,L)) .AND. (LIST.PIR(BP2,L)) .AND. (0 .NE. BP1-BP2)) .IMPLIES. ((LIST.PIR(BP1,SUBLIST(BP1,BP2,L))) .AND. (0 .EQ. DIFLR(BP1,BP1,L), .TRUE.) .AND. (LIST.PIR(BP2,L),LV(BP1,L),0 .NE. -BP1+BP2))

```
3                                                                                                                    1
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1     (SPACE(FP)-IRL .EQ. 0)
      3     S=S+1
      4     FP=SPACE(FP+1)                                                                                         2
            (FP-BP2 .EQ. 0)
      7     (S .EQ. 0)
      9     FAIL=0
      COND FAIL=0 .AND. LIST.PIR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1 .NE.
      COND BP2))
                                                                                                                     3
                                                                                  ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                                  -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                                  .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                                  (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                                  .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                                  .LE. S) .AND. (0 .EQ.
                                                                                  -IRL+SPACE(FP)) .AND. (0 .EQ.
                                                                                  -BP2+SPACE(1+FP)) .AND. (0 .EQ. 1+S))
                                                                                   .IMPLIES.
                                                                                  ((LIST.PIR(BP2,L)) .AND. (0 .EQ.
                                                                                  DIFLR(BP1,BP2,L),0 .NE. -BP1+BP2))

4
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1     (SPACE(FP)-IRL .EQ. 0)
      3     S=S+1
      4     FP=SPACE(FP+1)
            (FP-BP2 .EQ. 0)
      7      .NOT. (S .EQ. 0)
      8     FAIL=1
      COND FAIL=1 .AND. LIST.PIR(BP2,L) .AND. ((BP1 .EQ. BP2) .OR. ((DIFLR(BP1,BP2,L)
      COND .NE. 0) .OR. ((DIFLR(BP1,FP,L) .LT.
      COND 0),LIST.PIR(FP,SUBLIST(BP1,BP2,L))),(BP1 .NE. BP2)))
                                                                                  ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                                  -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                                  .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                                  (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                                  .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                                  .LE. S) .AND. (0 .EQ.
                                                                                  -IRL+SPACE(FP)) .AND. (0 .EQ.
                                                                                  -BP2+SPACE(1+FP)) .AND. (0 .NE. 1+S))
                                                                                   .IMPLIES.
                                                                                  ((LIST.PIR(BP2,L)) .AND. ((0 .EQ.
                                                                                  -BP1+BP2) .OR. ((0 .NE.
                                                                                  -DIFLR(BP1,BP2,L)) .OR. (0 .LT.
                                                                                  -DIFLR(BP1,SPACE(1+FP),L),LIST.PIR(SP
                                                                                  ACE(1+FP),SUBLIST(BP1,BP2,L)),0 .NE.
                                                                                  -BP1+BP2))))

5
```

```
COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
COND .GE. 0))
1      (SPACE(FP)-IRL .EQ. 0)
3      S=S+1
4      FP=SPACE(FP+1)
        .NOT. (FP-BP2 .EQ. 0)
5      (FP .NE. 0)
COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
COND .GE. 0))
                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                    .LE. S) .AND. (0 .EQ.
                                                    -IRL+SPACE(FP)) .AND. (0 .NE.
                                                    -BP2+SPACE(1+FP)) .AND. (0 .NE.
                                                    -SPACE(1+FP)))
                                                     .IMPLIES.
                                                    ((LIST.PIR(SPACE(1+FP),SUBLIST(BP1,BP
                                                    2,L))) .AND. (0 .EQ.
                                                    -1+DIFLR(BP1,SPACE(1+FP),L)-S,0 .LE.
                                                    1+S) .AND.
                                                    (LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                    -BP1+BP2))

6
COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
COND .GE. 0))
1      (SPACE(FP)-IRL .EQ. 0)
3      S=S+1
4      FP=SPACE(FP+1)
        .NOT. (FP-BP2 .EQ. 0)
5       .NOT. (FP .NE. 0)
9      FAIL=0
COND FAIL=0 .AND. LIST.PIR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1 .NE.
COND BP2))
                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                    .LE. S) .AND. (0 .EQ.
                                                    -IRL+SPACE(FP)) .AND. (0 .NE.
                                                    -BP2+SPACE(1+FP)) .AND. (0 .EQ.
                                                    -SPACE(1+FP)))
                                                     .IMPLIES.
                                                    ((LIST.PIR(BP2,L)) .AND. (0 .EQ.
                                                    DIFLR(BP1,BP2,L),0 .NE. -BP1+BP2))

7
```

```
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1      .NOT. (SPACE(FP)-IRL .EQ. 0)
      2      (SPACE(FP)-IRR .NE. 0)
      4      FP=SPACE(FP+1)
             (FP-BP2 .EQ. 0)
      7      (S .EQ. 0)
      9      FAIL=0
      COND FAIL=0 .AND. LIST.PIR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1 .NE.
      COND BP2))
                                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                    .LE. S) .AND. (0 .NE.
                                                                    -IRL+SPACE(FP)) .AND. (0 .NE.
                                                                    IRR-SPACE(FP)) .AND. (0 .EQ.
                                                                    -BP2+SPACE(1+FP)) .AND. (0 .EQ. S))
                                                                     .IMPLIES.
                                                                    ((LIST.PIR(BP2,L)) .AND. (0 .EQ.
                                                                    DIFLR(BP1,BP2,L),0 .NE. -BP1+BP2))

      8
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1      .NOT. (SPACE(FP)-IRL .EQ. 0)
      2      (SPACE(FP)-IRR .NE. 0)
      4      FP=SPACE(FP+1)
             (FP-BP2 .EQ. 0)
      7      .NOT. (S .EQ. 0)
      8      FAIL=1
      COND FAIL=1 .AND. LIST.PIR(BP2,L) .AND. ((BP1 .EQ. BP2) .OR. ((DIFLR(BP1,BP2,L)
      COND .LE. 0) .OR. ((DIFLR(BP1,FP,L) .LT.
      COND 0),LIST.PIR(FP,SUBLIST(BP1,BP2,L))),(BP1 .NE. BP2)))
                                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                    .LE. S) .AND. (0 .NE.
                                                                    -IRL+SPACE(FP)) .AND. (0 .NE.
                                                                    IRR-SPACE(FP)) .AND. (0 .EQ.
                                                                    -BP2+SPACE(1+FP)) .AND. (0 .NE. S))
                                                                     .IMPLIES.
                                                                    ((LIST.PIR(BP2,L)) .AND. ((0 .EQ.
                                                                    -BP1+BP2) .OR. ((0 .NE.
                                                                    -DIFLR(BP1,BP2,L)) .OR. (0 .LT.
                                                                    -DIFLR(BP1,SPACE(1+FP),L),LIST.PIR(SP
                                                                    ACE(1+FP),SUBLIST(BP1,BP2,L)),0 .NE.
                                                                    -BP1+BP2))))
```

```
9
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1      .NOT. (SPACE(FP)-IRL .EQ. 0)
      2     (SPACE(FP)-IRR .NE. 0)
      4     FP=SPACE(FP+1)
             .NOT. (FP-BP2 .EQ. 0)
      5     (FP .NE. 0)
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
                                                                  ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                  -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                  .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                  (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                  .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                  .LE. S) .AND. (0 .NE.
                                                                  -IRL+SPACE(FP)) .AND. (0 .NE.
                                                                  IRR-SPACE(FP)) .AND. (0 .NE.
                                                                  -BP2+SPACE(1+FP)) .AND. (0 .NE.
                                                                  -SPACE(1+FP)))
                                                                   .IMPLIES.
                                                                  ((LIST.PIR(SPACE(1+FP),SUBLIST(BP1,BP
                                                                  2,L))) .AND. (0 .EQ.
                                                                  DIFLR(BP1,SPACE(1+FP),L)-S,0 .LE. S)
                                                                  .AND. (LIST.PIR(BP2,L),LV(BP1,L),0
                                                                  .NE. -BP1+BP2))

10
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1      .NOT. (SPACE(FP)-IPL .EQ. 0)
      2     (SPACE(FP)-IRR .NE. 0)
      4     FP=SPACE(FP+1)
             .NOT. (FP-BP2 .EQ. 0)
      5      .NOT. (FP .NE. 0)
      9     FAIL=0
      COND FAIL=0 .AND. LIST.PIR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1 .NE.
      COND BP2))
                                                                  ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                  -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                  .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                  (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                  .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                  .LE. S) .AND. (0 .NE.
                                                                  -IRL+SPACE(FP)) .AND. (0 .NE.
                                                                  IRR-SPACE(FP)) .AND. (0 .NE.
                                                                  -BP2+SPACE(1+FP)) .AND. (0 .EQ.
                                                                  -SPACE(1+FP)))
                                                                   .IMPLIES.
                                                                  ((LIST.PIR(BP2,L)) .AND. (0 .EQ.
                                                                  DIFLR(BP1,BP2,L),0 .NE. -BP1+BP2))
```

```
11
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1       .NOT. (SPACE(FP)-IRL .EQ. 0)
      2       .NOT. (SPACE(FP)-IRR .NE. 0)
      6      S=S-1
             (S .LT. 0)
      8      FAIL=1
      COND FAIL=1 .AND. LIST.PIR(BP2,L) .AND. ((BP1 .EQ. BP2) .OR. ((DIFLR(BP1,BP2,L)
      COND .NE. 0) .OR. ((DIFLR(BP1,FP,L) .LT.
      COND 0),LIST.PIR(FP,SUBLIST(BP1,BP2,L))),(BP1 .NE. BP2)))
```

```
((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
-BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
.AND. (0 .EQ. -IRR+Q(42)) .AND.
(LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
.AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
.LE. S) .AND. (0 .NE.
-IRL+SPACE(FP)) .AND. (0 .EQ.
IRR-SPACE(FP)) .AND. (0 .LT. 1-S))
 .IMPLIES.
((LIST.PIR(BP2,L)) .AND. ((0 .EQ.
-BP1+BP2) .OR. ((0 .NE.
-DIFLR(BP1,BP2,L)) .OR. (0 .LT.
-DIFLR(BP1,FP,L),LIST.PIR(FP,SUBLIST(
BP1,BP2,L)),0 .NE. -BP1+BP2))))
```

```
12
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1       .NOT. (SPACE(FP)-IRL .EQ. 0)
      2       .NOT. (SPACE(FP)-IRR .NE. 0)
      6      S=S-1
             .NOT. (S .LT. 0)
      4      FP=SPACE(FP+1)
             (FP-BP2 .EQ. 0)
      7      (S .EQ. 0)
      9      FAIL=0
      COND FAIL=0 .AND. LIST.PIR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1 .NE.
      COND BP2))
```

```
((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
-BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
.AND. (0 .EQ. -IRR+Q(42)) .AND.
(LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
.AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
.LE. S) .AND. (0 .NE.
-IRL+SPACE(FP)) .AND. (0 .EQ.
IRR-SPACE(FP)) .AND. (0 .LE. -1+S)
.AND. (0 .EQ. -BP2+SPACE(1+FP))
.AND. (0 .EQ. -1+S))
 .IMPLIES.
((LIST.PIR(BP2,L)) .AND. (0 .EQ.
DIFLR(BP1,BP2,L),0 .NE. -BP1+BP2))
```

```
13
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1       .NOT. (SPACE(FP)-IRL .EQ. 0)
      2       .NOT. (SPACE(FP)-IRR .NE. 0)
      6      S=S-1
              .NOT. (S .LT. 0)
      4      FP=SPACE(FP+1)
             (FP-BP2 .EQ. 0)
      7       .NOT. (S .EQ. 0)
      8      FAIL=1
      COND FAIL=1 .AND. LIST.PIR(BP2,L) .AND. ((BP1 .EQ. BP2) .OR. ((DIFLR(BP1,BP2,L)
      COND .NE. 0) .OR. ((DIFLR(BP1,FP,L) .LT.
      COND 0)),LIST.PIR(FP,SUBLIST(BP1,BP2,L))),(BP1 .NE. BP2)))
                                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                    .LE. S) .AND. (0 .NE.
                                                                    -IRL+SPACE(FP)) .AND. (0 .EQ.
                                                                    IRR-SPACE(FP)) .AND. (0 .LE. -1+S)
                                                                    .AND. (0 .EQ. -BP2+SPACE(1+FP))
                                                                    .AND. (0 .NE. -1+S))
                                                                     .IMPLIES.
                                                                    ((LIST.PIR(BP2,L)) .AND. ((0 .EQ.
                                                                    -BP1+BP2) .OR. ((0 .NE.
                                                                    -DIFLR(BP1,BP2,L)) .OR. (0 .LT.
                                                                    -DIFLR(BP1,SPACE(1+FP),L),LIST.PIR(SP
                                                                    ACE(1+FP),SUBLIST(BP1,BP2,L)),0 .NE.
                                                                    -BP1+BP2))))

14
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1       .NOT. (SPACE(FP)-IRL .EQ. 0)
      2       .NOT. (SPACE(FP)-IRR .NE. 0)
      6      S=S-1
      3       .NOT. (S .LT. 0)
      4      FP=SPACE(FP+1)
              .NOT. (FP-BP2 .EQ. 0)
      5      (FP .NE. 0)
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      2 COND .GE. 0))
                                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                    .LE. S) .AND. (0 .NE.
      1
```

```
                                                                    -IRL+SPACE(FP)) .AND. (0 .EQ.                    1
                                                                    IRR-SPACE(FP)) .AND. (0 .LE. -1+S)
                                                                    .AND. (0 .NE. -BP2+SPACE(1+FP))
                                                                    .AND. (0 .NE. -SPACE(1+FP)))
                                                                     .IMPLIES.
                                                                    ((LIST.PIR(SPACE(1+FP),SUBLIST(BP1,BP
                                                                    2,L))) .AND. (0 .EQ.                             2
                                                                    1+DIFLR(BP1,SPACE(1+FP),L)-S,0 .LE.
                                                                    -1+S) .AND.
                                                                    (LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                    -BP1+BP2))

                                                                                                                     3
15
      COND (LIST.PIR(BP2,L),LV(BP1,L),(BP1 .NE. BP2)) .AND. IRL=Q(41) .AND. IRR=Q(42)
      COND .AND. LIST.PIR(FP,SUBLIST(BP1,BP2,L)) .AND. ((S .EQ. DIFLR(BP1,FP,L)),(S
      COND .GE. 0))
      1       .NOT. (SPACE(FP)-IRL .EQ. 0)
      2       .NOT. (SPACE(FP)-IRR .NE. 0)
      6      S=S-1
              .NOT. (S .LT. 0)
      4      FP=SPACE(FP+1)
              .NOT. (FP-BP2 .EQ. 0)
      5       .NOT. (FP .NE. 0)
      9      FAIL=0
      COND FAIL=0 .AND. LIST.PIR(BP2,L) .AND. ((DIFLR(BP1,BP2,L) .EQ. 0),(BP1 .NE.
      COND BP2))
                                                                    ((LIST.PIR(BP2,L),LV(BP1,L),0 .NE.
                                                                    -BP1+BP2) .AND. (0 .EQ. -IRL+Q(41))
                                                                    .AND. (0 .EQ. -IRR+Q(42)) .AND.
                                                                    (LIST.PIR(FP,SUBLIST(BP1,BP2,L)))
                                                                    .AND. (0 .EQ. DIFLR(BP1,FP,L)-S,0
                                                                    .LE. S) .AND. (0 .NE.
                                                                    -IRL+SPACE(FP)) .AND. (0 .EQ.
                                                                    IRR-SPACE(FP)) .AND. (0 .LE. -1+S)
                                                                    .AND. (0 .NE. -BP2+SPACE(1+FP))
                                                                    .AND. (0 .EQ. -SPACE(1+FP)))
                                                                     .IMPLIES.
                                                                    ((LIST.PIR(BP2,L)) .AND. (0 .EQ.
                                                                    DIFLR(BP1,BP2,L),0 .NE. -BP1+BP2))
```

2.3) BAL2(BP1,BP2)

This subroutine checks whether the expression between the pointers BP1 and up to but not including BP2 IS Balanced or not.

VERIFICATION:

Path 1: BP1=BP2 has not been changed.

Path 2: The conclusion holds by definition.

Path 3: Since DIFLR(BP1,FP,I)=S, S=-1, SPACE(FP)=IRL, IRL=Q(41), SPACE(FP+1)=BP2, hence DIFLR(BP1,BP2,L)=0.

Path 4: SIMILAR TO THE ABOVE path, DIFLR(BP1,SPACE(FP+1),L) .LT. 0.

Path 5: Simliar to the above path, DIFLR(BP1,SPACE(FP+1),L)-S-1=0.

Path 6: Similar to path 3.

Path 7: Similar to path 6.

Path 8: Similar to path 4.

Path 9: Similar to path 5.

Path 10: Similar to path 3.

Path 11: Since SPACE(FP)≠IRL and SPACE(FP)=IRR=Q(42), by definition, DIFLR(BP1,BP2,L)≠0 and DIFLR(BP1,FP,L) .LT. 0

Path 12: Similar to path 3.

Path 13: Similar to path 11.

Path 14: Similar to path 5.

Path 15: Similar to path 3.

```
CC     LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC     SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC     .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC     LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC     CDRN(I,L)=CDR(CDRN(I-1,L)).
CC     CDRN(0,L)=L.
CC     ELT(I,L)=CAR(CDRN(I-1,L)).
CC     ALL(F(X),X,I,J) .IN. (F(X), X.IN.(I..J))
CC     ALL(I,L1,(A,X),J,L2,(B,Y))=ALL(ALL(ELT(I,L1) .NE. ELT(J,L2),
CC     J,B,Y),I,A,X).
CCND    LV(STRING,L1),LV(BCHARS,L2)
       SUBROUTINE BREAK (STRING,BCHARS,PATTER)
       COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
       INTEGER SPACE,FAIL,PATTER,STRING,SCHAR,BCHAR,BCHARS,BADDR
C  THIS SUBROUTINE EXAMINES THE CHARACTERS IN THE LIST STRING FOR A
C  MATCH WITH ANY OF THE CHARACTERS IN THE LIST BCHARS. IF A MATCH
C  OCCURS, PATTERN POINTS TO THE LOCATION IN STRING WHERE THE FIRST
C  MATCH OCCURS AND FAIL IS SET TO 0. IF NO MATCH IS FOUND, FAIL IS SET
C  TO 1 AND PATTERN IS ASSIGNED THE NULL STRING. (THE PARAMETERS STRING
C  AND BCHARS ARE NOT CHANGED.)
       FAIL=0
C  IF STRING IS NULL, THE STATEMENT FAILS AND PATTERN IS THE NULL STRING
       IF(STRING) 7,5,7
    7  PATTER=0
       IPTR=STRING
C  PULL ONE CHARACTER OUT OF STRING
    1  SCHAR=SPACE(IPTR)
       BADDR=BCHARS
C  PULL ONE CHARACTER OUT OF THE LIST OF BREAKING CHARACTERS
    2  BCHAR=SPACE(BADDR)
C  SAVE THE LAST CHARACTER ADDRESS OF STRING
C  TEST FOR A MATCH
       IF(BCHAR-SCHAR) 3,6,3
CCND    FAIL=0,LIST.PIR(IPTR,L1),LIST.PIR(BADDR,L2),LV(BCHARS,L2),
CCND    ALL(ALL(ELT(I,L1) .NE. ELT(J,L2),J,1,LENGTH(L2)),I,1,
CCND    LENGTH(SUBLIST(STRING,IPTR,L1))),SCHAR=SPACE(IPTR),
CCND    ALL((SPACE(IPTR) .NE. ELT(J,L2)),J,1,
CCND    LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
C  GET THE ADDRESS OF THE NEXT BREAKING CHARACTER
    3  BADDR=SPACE(BADDR+1)
       IF(BADDR) 2,4,2
    4  PATTER=IPTR
C  GET THE ADDRESS OF THE NEXT CHARACTER IN STRING
       IPTR=SPACE(IPTR+1)
C  TEST FOR THE END OF THE LIST STRING
       IF(IPTR) 1,5,1
    5  FAIL=1
       PATTER=0
CCND    FAIL=1,
CCND    ALL(ALL(ELT(I,L1) .NE. ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))
       RETURN
6      CONTINUE
CCND    FAIL=0,ALL(ALL(ELT(I,L1) .NE. ELT(J,L2),J,1,
CCND    LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))),
```

```
COND    LIST.PIP(IPTR,L1),EXISTS((SPACE(IPTR)=ELT(J,L2)),J,1,LENGTH(L2))
        RETURN
        END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(STRING,L1) .AND. LV(BCHARS,L2)		
SUBROUTINE BREAK (STRING,BCHARS,PATTER)		
COMMON/BLOCK/SPACE(10000),FAIL,IFOO		
INTEGER SPACE,FAIL,PATTER,STRING,SCHAR,BCHAR,BCHARS,BADDR		
FAIL=0	[FAIL]	[]
IF(STRING .EQ. 0)GO TO 5		
7 PATTER=0	[PATTER]	[]
IPTR=STRING	[IPTR]	[STRING]
1 SCHAR=SPACE(IPTR)	[SCHAR]	[SPACE,IPTR]
BADDR=BCHARS	[BADDR]	[BCHARS]
2 BCHAR=SPACE(BADDR)	[BCHAR]	[SPACE,BADDR]
IF(BCHAR-SCHAR .EQ. 0)GO TO 6		
COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.		
COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.		
COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1)))		
COND .AND. SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.		
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))		
3 BADDR=SPACE(BADDR+1)	[BADDR]	[SPACE,BADDR]
IF(BADDR .NE. 0)GO TO 2		
4 PATTER=IPTR	[PATTER]	[IPTR]
IPTR=SPACE(IPTR+1)	[IPTR]	[SPACE,IPTR]
IF(IPTR .NE. 0)GO TO 1		
5 FAIL=1	[FAIL]	[]
PATTER=0	[PATTER]	[]
COND FAIL=1 .AND. ALL(ALL(ELT(I,L1) .NE.		
COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))		
RETURN		
6 CONTINUE		
COND FAIL=0 .AND. ALL(ALL(ELT(I,L1) .NE.		
COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1)))		
COND .AND. LIST.PIR(IPTR,L1) .AND. EXISTS((SPACE(IPTR) .EQ.		
COND ELT(J,L2)),J,1,LENGTH(L2))		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(STRING,L1) .AND. LV(BCHARS,L2) SUBROUTINE BREAK (STRING,BCHARS,PATTER) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SPACE,FAIL,PATTER,STRING,SCHAR,BCHAR,BCHARS,BADDR FAIL=0 (STRING .EQ. 0) 5 FAIL=1 PATTER=0 COND FAIL=1 .AND. ALL(ALL(ELT(I,L1) .NE. COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))	((LV(STRING,L1)) .AND. (LV(BCHARS,L2)) .AND. (0 .EQ. STRING)) .IMPLIES. ((ALL(ALL(0 .NE. -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)), I,1,LENGTH(L1))))
2	COND LV(STRING,L1) .AND. LV(BCHARS,L2) SUBROUTINE BREAK (STRING,BCHARS,PATTER) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SPACE,FAIL,PATTER,STRING,SCHAR,BCHAR,BCHARS,BADDR FAIL=0 .NOT. (STRING .EQ. 0) 7 PATTER=0 IPTR=STRING 1 SCHAR=SPACE(IPTR) BADDR=BCHARS 2 BCHAR=SPACE(BADDR) (BCHAR-SCHAR .EQ. 0) 6 CONTINUE COND FAIL=0 .AND. ALL(ALL(ELT(I,L1) .NE. COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND. COND LIST.PIR(IPTR,L1) .AND. EXISTS((SPACE(IPTR) .EQ. ELT(J,L2)),J,1,LENGTH(L2))	((LV(STRING,L1)) .AND. (LV(BCHARS,L2)) .AND. (0 .NE. STRING) .AND. (0 .EQ. SPACE(BCHARS)-SPACE(STRING))) .IMPLIES. ((LIST.PIR(STRING,L1)) .AND. (EXISTS(0 .EQ. ELT(J,L2)-SPACE(STRING),J,1,LENGTH(L2))) .AND. (ALL(ALL(0 .NE. -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)), I,1,LENGTH(SUBLIST(STRING,STRING,L1)))))

```
                                                                                                    3
      COND LV(STRING,L1) .AND. LV(BCHARS,L2)
           SUBROUTINE BREAK (STRING,BCHARS,PATTER)
           COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
           INTEGER SPACE,FAIL,PATTER,STRING,SCHAR,BCHAR,BCHARS,BADDR
           FAIL=0
            .NOT. (STRING .EQ. 0)
      7    PATTER=0
           IPTR=STRING
      1    SCHAR=SPACE(IPTR)
           BADDR=BCHARS
      2    BCHAR=SPACE(BADDR)
            .NOT. (BCHAR-SCHAR .EQ. 0)
      COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
      COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
                                                                          ((LV(STRING,L1)) .AND.
                                                                          (LV(BCHARS,L2)) .AND. (0 .NE.
                                                                          STRING) .AND. (0 .NE.
                                                                          SPACE(BCHARS)-SPACE(STRING)))
                                                                           .IMPLIES.
                                                                          ((LIST.PIR(STRING,L1)) .AND.
                                                                          (LIST.PIR(BCHARS,L2)) .AND.
                                                                          (LV(BCHARS,L2)) .AND. (ALL(0 .NE.
                                                                          ELT(J,L2)-SPACE(STRING),J,1,LENGTH(SU
                                                                          BLIST(BCHARS,SPACE(1+BCHARS),L2))))
                                                                          .AND. (ALL(ALL(0 .NE.
                                                                          -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                          I,1,LENGTH(SUBLIST(STRING,STRING,L1))
                                                                          )))

      4
      COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
      COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
      3      BADDR=SPACE(BADDR+1)
             (BADDR .NE. 0)
      2      BCHAR=SPACE(BADDR)
      3      (BCHAR-SCHAR .EQ. 0)
      6      CONTINUE
      COND FAIL=0 .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND LIST.PIR(IPTR,L1) .AND. EXISTS((SPACE(IPTR) .EQ. ELT(J,L2)),J,1,LENGTH(L2))
                                                                          ((0 .EQ. FAIL) .AND.
                                                                          (LIST.PIR(IPTR,L1)) .AND.
                                                                          (LIST.PIR(BADDR,L2)) .AND.
                                                                          (LV(BCHARS,L2)) .AND. (ALL(ALL(0
                                                                          .NE.
                                                                          -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                          I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
                                                                           .AND. (0 .EQ. -SCHAR+SPACE(IPTR)))
```

```
                                                                .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(SUBL
                                                                IST(BCHARS,SPACE(1+BADDR),L2))))
                                                                .AND. (0 .NE. -SPACE(1+BADDR)) .AND.
                                                                (0 .EQ.
                                                                -SCHAR+SPACE(SPACE(1+BADDR))))
                                                                 .IMPLIES.
                                                                ((LIST.PIR(IPTR,L1)) .AND. (EXISTS(0
                                                                .EQ.
                                                                ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(L2))
                                                                ) .AND. (ALL(ALL(0 .NE.
                                                                -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
                                                                )

5
      COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
      COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
      3     BADDR=SPACE(BADDR+1)
            (BADDR .NE. 0)
      2     BCHAR=SPACE(BADDR)
             .NOT. (BCHAR-SCHAR .EQ. 0)
      COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
      COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
                                                                ((0 .EQ. FAIL) .AND.
                                                                (LIST.PIR(IPTR,L1)) .AND.
                                                                (LIST.PIR(BADDR,L2)) .AND.
                                                                (LV(BCHARS,L2)) .AND. (ALL(ALL(0
                                                                .NE.
                                                                -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
                                                                 .AND. (0 .EQ. -SCHAR+SPACE(IPTR))
                                                                .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(SUBL
                                                                IST(BCHARS,SPACE(1+BADDR),L2))))
                                                                .AND. (0 .NE. -SPACE(1+BADDR)) .AND.
                                                                (0 .NE.
                                                                -SCHAR+SPACE(SPACE(1+BADDR))))
                                                                 .IMPLIES.
                                                                ((LIST.PIR(IPTR,L1)) .AND.
                                                                (LIST.PIR(SPACE(1+BADDR),L2)) .AND.
                                                                (LV(BCHARS,L2)) .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(SUBL
                                                                IST(BCHARS,SPACE(1+SPACE(1+BADDR)),L2
                                                                )))) .AND. (ALL(ALL(0 .NE.
                                                                -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
                                                                )
```

```
6
      COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
      COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
      3     BADDR=SPACE(BADDR+1)
             .NOT. (BADDR .NE. 0)
      4     PATTER=IPTR
            IPTR=SPACE(IPTR+1)
            (IPTR .NE. 0)
      1     SCHAR=SPACE(IPTR)
            BADDR=BCHARS
      2     BCHAR=SPACE(BADDR)
            (BCHAR-SCHAR .EQ. 0)
      6     CONTINUE
      COND FAIL=0 .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND LIST.PIR(IPTR,L1) .AND. EXISTS((SPACE(IPTR) .EQ. ELT(J,L2)),J,1,LENGTH(L2))
                                                                  ((0 .EQ. FAIL) .AND.
                                                                  (LIST.PIR(IPTR,L1)) .AND.
                                                                  (LIST.PIR(BADDR,L2)) .AND.
                                                                  (LV(BCHARS,L2)) .AND. (ALL(ALL(0
                                                                  .NE.
                                                                  -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                  I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
                                                                   .AND. (0 .EQ. -SCHAR+SPACE(IPTR))
                                                                  .AND. (ALL(0 .NE.
                                                                  ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(SUBL
                                                                  IST(BCHARS,SPACE(1+BADDR),L2))))
                                                                  .AND. (0 .EQ. -SPACE(1+BADDR)) .AND.
                                                                  (0 .NE. -SPACE(1+IPTR)) .AND. (0
                                                                  .EQ.
                                                                  SPACE(BCHARS)-SPACE(SPACE(1+IPTR))))
                                                                   .IMPLIES.
                                                                  ((LIST.PIR(SPACE(1+IPTR),L1)) .AND.
                                                                  (EXISTS(0 .EQ.
                                                                  ELT(J,L2)-SPACE(SPACE(1+IPTR)),J,1,LE
                                                                  NGTH(L2))) .AND. (ALL(ALL(0 .NE.
                                                                  -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                                  I,1,LENGTH(SUBLIST(STRING,SPACE(1+IPT
                                                                  R),L1))))))
3

7
      COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
      COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
      COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
      COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
      3     BADDR=SPACE(BADDR+1)
             .NOT. (BADDR .NE. 0)
      4     PATTER=IPTR
            IPTR=SPACE(IPTR+1)
            (IPTR .NE. 0)
```

```
1       SCHAR=SPACE(IPTR)
        BADDR=BCHARS
2       BCHAR=SPACE(BADDR)
         .NOT. (BCHAR-SCHAR .EQ. 0)
COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
                                                    ((0 .EQ. FAIL) .AND.
                                                    (LIST.PIR(IPTR,L1)) .AND.
                                                    (LIST.PIR(BADDR,L2)) .AND.
                                                    (LV(BCHARS,L2)) .AND. (ALL(ALL(0
                                                    .NE.
                                                    -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                    I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
                                                     .AND. (0 .EQ. -SCHAR+SPACE(IPTR))
                                                    .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(SUBL
                                                    IST(BCHARS,SPACE(1+BADDR),L2))))
                                                    .AND. (0 .EQ. -SPACE(1+BADDR)) .AND.
                                                    (0 .NE. -SPACE(1+IPTR)) .AND. (0
                                                    .NE.
                                                    SPACE(BCHARS)-SPACE(SPACE(1+IPTR))))
                                                     .IMPLIES.
                                                    ((LIST.PIR(SPACE(1+IPTR),L1)) .AND.
                                                    (LIST.PIR(BCHARS,L2)) .AND.
                                                    (LV(BCHARS,L2)) .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SPACE(1+IPTR)),J,1,LE
                                                    NGTH(SUBLIST(BCHARS,SPACE(1+BCHARS),L
                                                    2)))) .AND. (ALL(ALL(0 .NE.
                                                    -ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
                                                    I,1,LENGTH(SUBLIST(STRING,SPACE(1+IPT
                                                    R),L1))))))

3
        COND FAIL=0 .AND. LIST.PIR(IPTR,L1) .AND. LIST.PIR(BADDR,L2) .AND.
        COND LV(BCHARS,L2) .AND. ALL(ALL(ELT(I,L1) .NE.
        COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) .AND.
        COND SCHAR=SPACE(IPTR) .AND. ALL((SPACE(IPTR) .NE.
        COND ELT(J,L2)),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2)))
        3       BADDR=SPACE(BADDR+1)
                 .NOT. (BADDR .NE. 0)
        4       PATTER=IPTR
                IPTR=SPACE(IPTR+1)
                 .NOT. (IPTR .NE. 0)
        5       FAIL=1
                PATTER=0
        COND FAIL=1 .AND. ALL(ALL(ELT(I,L1) .NE.
        COND ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(L1))
                                                    ((0 .EQ. FAIL) .AND.
                                                    (LIST.PIR(IPTR,L1)) .AND.
                                                    (LIST.PIR(BADDR,L2)) .AND.
                                                    (LV(BCHARS,L2)) .AND. (ALL(ALL(0
                                                    .NE.
```

```
-ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
I,1,LENGTH(SUBLIST(STRING,IPTR,L1))))
 .AND. (0 .EQ. -SCHAR+SPACE(IPTR))
.AND. (ALL(0 .NE.
ELT(J,L2)-SPACE(IPTR),J,1,LENGTH(SUBL
IST(BCHARS,SPACE(1+BADDR),L2))))
.AND. (0 .EQ. -SPACE(1+BADDR)) .AND.
(0 .EQ. -SPACE(1+IPTR)))
 .IMPLIES.
((ALL(ALL(0 .NE.
-ELT(I,L1)+ELT(J,L2),J,1,LENGTH(L2)),
I,1,LENGTH(L1))))
```

2.4) BREAK(STRING,BCHARS,PATTR)

STRING = subject string
BCHARS = string of break characters
PATTR = points to the character in STRING where the sucessful match to one of the break characters was made.

This works like the BREAK function of SNOBOL.

example: STRING ="VOWELS"
result : FAIL is set to 0 and PATTR points to the "O" in STRING.

VERIFICATION:

Path 1: Since STRING=0, hence L1 is nil and the conclusion is true by definition.

Path 2: LV(STRING,L1) implies LIST.PIR(STRING,L1). Since SPACE(BCHARS)=SPACE(STRING) and LV(BCHARS,L2) implies SPACE(BCHARS) is some ELT(J,L2),J,1, LENGTH(L2), hence SPACE(STRING)=ELT(J,L2),J,1, LENGTH(L2). Since SUBLIST(STRING,STRING,L1) is nil, the rest is true by definition.

Path 3: By using the same arguments as above this path can be verified to be true,

Path 4: LIST.PIR(IPTR,L1) is true through out the whole path. EXISTS(ELT(J,L2)≠SPACE(IPTR),J,1,LENGTH(L2)) is true since ALL(ELT(J,L2)≠SPACE(IPTR),J,1, LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2))) is true. Finally, ALL(ALL(ELT(I,L1)≠ELT(J,L2),J,1,LENGTH(L2)) I,1,LENGTH(SUBLIST(STRING,IPTR,L1)) remains true throughout the whole path.

Path 5: Since LIST.PIR(BADDR,L2), hence LIST.PIR(SPACE (BADDR+1),L2). ALL(ALL(ELT(I,L1)≠ELT(J,L2),J,1, length(L2)),I,1,length(sublist(STRING,IPTR,L1))) has not been changed in this path. ALL(ELT(J,L2)≠SPACE(IPTR),J,1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2))) and SPACE(IPTR)= SCHAR and SCHAR≠SPACE(SPACE(BADDR+1)) implies ALL(ELT(J,L2)≠SPACE(IPTR),J,1,LENGTH(SUBLIST(BCHARS,SPACE(SPACE(BADDR+1)+1),L2))).

Path 6: Since SPACE(SPACE(IPTR+1))=SPACE(BCHARS) and LV(BCHARS,L2) hence EXISTS(SPACE(SPACE(IPTR+1))=ELT(J,L2),J1,LENGTH(L2)).
ALL(ALL(ELT(I,L1)≠ELT(J,L2),J1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,IPTR,L1))) and
ALL(ELT(J,L2)≠SPACE(IPTR),J1,LENGTH(SUBLIST(BCHARS,SPACE(BADDR+1),L2))) and SPACE(BADDR+1)=0 implies
ALL(ALL(ELT(I,L1)≠ELT(J,L2),J,1,LENGTH(L2)),I,1,LENGTH(SUBLIST(STRING,SPACE(IPTR+1),L1))).

Path 7: This path is similar to the above one except that SPACE(SPACE(IPTR+1))≠SPACE(BCHARS).

Path 8: This path is also similar to the above two paths. Since SPACE(BADDR+1)=0 and SPACE(IPTR+1)=0, all elements in L1 have been checked against all elements in L2, hence ALL(ALL(ELT(I,L1)≠ELT(J,L2), J,1,LENGTH(L2)),I,1,LENGTH(L1)).

```
CC  LENGTH(L) = (IF L THEN 0 ELSE LENGTH(CDR(L))+1)
CC  ALL(F(X),X,I,J) .IN. (F(X), X.IN.(I..J))
CC  EXISTS(F(X),X,I,J) .IN. (F(X),I.LE.X,X.LE.J)
CC  LIST.PTR(P,L) .IN. EXISTS(P=CDR(X,L), X,1,LENGTH(L))
C   LV(LP,L) MEANS THE LIST VALUE OF LP IS L
CC  LV(LP,L) = ((LP=0,L=0)), ((1.LE.K(I), K(I).LE.5000),
CC  SPACE(2**K(I)-1) = L(I), SPACE(2**K(I)) = 2**K(I+1) -1),
CC  I .IN. (1..N)), LP = 2**K(1) - 1)
C   COMP(L2,L3) IS THE COMPOSITION OF LISTS L2 AND L3
CC  L=COMP(L2,L3) .IN. (ALL(ELT(K,L)=ELT(K,L2),K,1,LENGTH(L2)).OR.
CC  ALL(ELT(K,L)=ELT(K,L3),K,LENGTH(L2),LENGTH(L)), LENGTH(L)=
CC  LENGTH(L2)+LENGTH(L3))
CC  ERROR(CONCAT) = #ERROR#
COND  LV(STR1,L1), LV(STR2,L2), LV(STR3,L3)
      SUBROUTINE CONCAT(STR1,STR2,STR3)
      COMMON/BLOCK/SPACE(10000),FAIL,IFOO
      INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL
C        TO SIMULATE THE SNOBOL STATEMENT                    USE FORTRAN
C  THIS SUBROUTINE CONCATENATES STRING STR2 TO STRING STR3 AND PUTS THE
C  RESULT IN STR1. ( STR2 AND STR3 ARE NOT CHANGED.)      EXAMPLES,
C                  Y = A B                                CALL CONCATE(Y,A,B)
C                  A = A B                                CALL CONCATE(A,A,B)
C                  A = A A                                CALL CONCATE(A,A,A)
C                  B = A B                                CALL CONCATE(B,A,B)
C                  Y = A B C                              CALL CONCATE(Y,A,B)
C                                                         CALL CONCATE(Y,Y,C)
C
C
      TEMP1=0
      IF(STR1) 600,500,200
500   IF(STR2) 600,105,200
COND  STR1=0, STR2=0, LV(STR1,L1),LV(STR2,L2),LV(STR3,L3)
  105 CALL COPY(STR1,STR3,0)
COND  LV(STR1,L1), LV(STR3,L3), STR1=STR3, STR2=0
      RETURN
  200 CONTINUE
COND  ((STR1.NE.0).OR.(STR2.NE.0)),LV(STR1,L1),LV(STR2,L2),LV(STR3,L3)
C  COPY STRING STR3 INTO STRING TEMP1
      CALL COPY(TEMP1,STR3,0)
COND  ((STR1.NE.0).OR.(STR2.NE.0)),LV(STR1,L1),LV(STR2,L2),LV(STR3,L3),
COND  LV(TEMP1,L3),TEMP1=STR3
C  IF STR1=STR2, SKIP NEXT PART
      IF(STR1-STR2) 1,6,1
C  COPY STRING STR2 INTO STRING STR1
COND  ((STR1.NE.0).OR.(STR2.NE.0)),LV(STR1,L1),LV(STR2,L2),LV(STR3,L3),
COND  LV(TEMP1,L3),TEMP1=STR3,(STR1.NE.STR2)
1     CALL COPY(STR1,STR2,0)
COND  STR1=STR2, LV(STR1,L),LV(STR3,L3),LV(STR2,L),LV(TEMP1,L3),
COND  TEMP1=STR3
C
C  FIND THE ADDRESS THAT CONTAINS THE ZERO POINTER OF STRING STR1
C
      IF(STR1) 2,6,2
2     STR1=TEMP1
COND  LV(STR1,L1), LV(STR3,L3), STR1=STR3, STR2=0
      RETURN
```

```
      2 FINDLN=STR1+1
COND  LV(STR1,L),LV(STR2,L2),LV(TEMP1,L3),LV(STR3,L3),
COND  STR1=STR2, TEMP1=STR3, LIST=IF(FINDLN-1,L)
      3 POINTE=SPACE(FINDLN)
        IF(POINTE) 4,5,4
      4 FINDLN=POINTE+1
        GO TO 2
C  CHANGE THE POINTER IN THE END OF STRING STR1 TO THE BEGINING OF
C  STRING STR3
      5 SPACE(FINDLN)=TEMP1
COND  LV(STR1,LIST), LIST=COMP(L2,L3)
        RETURN
6000      PRINT 6000
6000    FORMAT(          /1X30HERROR IN ARG TO CONCATE                    )
COND  ERROR(CONCAT), STR1.LT.0 .OR. STR2.LT.0 .OR. STR3.LT.0
        RETURN
        END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3)		
SUBROUTINE CONCAT(STR1,STR2,STR3)		
COMMON/BLOCK/SPACE(10000),FAIL,IFOO		
INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL		
TEMP1=0		
IF(STR1 .LT. 0)GO TO 600	[TEMP1]	[]
IF(STR1 .GT. 0)GO TO 200		
500 IF(STR2 .LT. 0)GO TO 600		
IF(STR2 .GT. 0)GO TO 200		
COND STR1=0 .AND. STR2=0 .AND. LV(STR1,L1) .AND. LV(STR2,L2) .AND.		
COND LV(STR3,L3)		
100 CALL COPY(STR1,STR3,0)		
COND LV(STR1,L1) .AND. LV(STR3,L3) .AND. STR1=STR3 .AND. STR2=0		
RETURN		
200 CONTINUE		
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND.		
COND LV(STR2,L2) .AND. LV(STR3,L3)		
CALL COPY(TEMP1,STR3,0)		
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND.		
COND LV(STR2,L2) .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3		
IF(STR1-STR2 .EQ. 0)GO TO 2		
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND.		
COND LV(STR2,L2) .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3		
COND .AND. (STR1 .NE. STR2)		
1 *CALL COPY(STR1,STR2,0)		
COND STR1=STR2 .AND. LV(STR1,L) .AND. LV(STR3,L3) .AND. LV(STR2,L)		
COND .AND. LV(TEMP1,L3) .AND. TEMP1=STR3		
IF(STR1 .NE. 0)GO TO 2		
5 STR1=TEMP1	[STR1]	[TEMP1]
COND LV(STR1,L1) .AND. LV(STR3,L3) .AND. STR1=STR3 .AND. STR2=0		
RETURN		
2 FINDEN=STR1+1	[FINDEN]	[STR1]
COND LV(STR1,L) .AND. LV(STR2,L2) .AND. LV(TEMP1,L3) .AND. LV(STR3,L3)		
COND .AND. STR1=STR2 .AND. TEMP1=STR3 .AND. LIST.PIR(FINDEN-1,L)		
3 POINTE=SPACE(FINDEN)	[POINTE]	[SPACE,FINDEN]
IF(POINTE .EQ. 0)GO TO 5		
4 FINDEN=POINTE+1	[FINDEN]	[POINTE]
GO TO 3		
5 SPACE(FINDEN)=TEMP1	[SPACE]	[SPACE,FINDEN,TEMP1]
COND LV(STR1,LIST) .AND. LIST=COMP(L2,L3)		
RETURN		
600 PRINT 6000		
6000 FORMAT (' /1X3GHERROR IN ARG TO CONCATE ')		
COND ERROR(CONCAT) .AND. STR1 .LT. 0 .OR. STR2 .LT. 0 .OR. STR3 .LT. 0		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3) SUBROUTINE CONCAT(STR1,STR2,STR3) COMMON/BLOCK/SPACE(10000),FAIL,IFOO INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL TEMP1=0 (STR1 .LT. 0) 6000 PRINT 6000 6000 FORMAT(/1X30HERROR IN ARG TO CONCATE) COND ERROR(CONCAT) .AND. STR1 .LT. 0 .OR. STR2 .LT. 0 .OR. STR3 .LT. 0	((LV(STR1,L1)) .AND. (LV(STR2,L2)) .AND. (LV(STR3,L3)) .AND. (0 .LT. -STR1)) .IMPLIES. (((ERROR(CONCAT)) .AND. (0 .LT. -STR1)) .OR. (0 .LT. -STR2)) .OR. (0 .LT. -STR3))
2	COND LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3) SUBROUTINE CONCAT(STR1,STR2,STR3) COMMON/BLOCK/SPACE(10000),FAIL,IFOO INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL TEMP1=0 .NOT. (STR1 .LT. 0) (STR1 .GT. 0) 200 CONTINUE COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2) COND .AND. LV(STR3,L3)	((LV(STR1,L1)) .AND. (LV(STR2,L2)) .AND. (LV(STR3,L3)) .AND. (0 .LE. STR1) .AND. (0 .LT. STR1)) .IMPLIES. ((LV(STR1,L1)) .AND. (LV(STR2,L2)) .AND. (LV(STR3,L3)) .AND. ((0 .NE. -STR1) .OR. (0 .NE. -STR2)))
3	COND LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3) SUBROUTINE CONCAT(STR1,STR2,STR3) COMMON/BLOCK/SPACE(10000),FAIL,IFOO INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL TEMP1=0 .NOT. (STR1 .LT. 0) .NOT. (STR1 .GT. 0) 500 (STR2 .LT. 0) 600 PRINT 6000 6000 FORMAT(/1X30HERROR IN ARG TO CONCATE) COND ERROR(CONCAT) .AND. STR1 .LT. 0 .OR. STR2 .LT. 0 .OR. STR3 .LT. 0	

```
                                                                ((LV(STR1,L1)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(STR3,L3)) .AND. (0 .LE.
                                                                STR1) .AND. (0 .LE. -STR1) .AND. (0
                                                                .LT. -STR2))
                                                                 .IMPLIES.
                                                                (((ERROR(CONCAT)) .AND. (0 .LT.
                                                                -STR1)) .OR. (0 .LT. -STR2)) .OR. (0
                                                                .LT. -STR3))

4
      COND LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3)
           SUBROUTINE CONCAT(STR1,STR2,STR3)
           COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
           INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL
           TEMP1=0
            .NOT. (STR1 .LT. 0)
            .NOT. (STR1 .GT. 0)
      500   .NOT. (STR2 .LT. 0)
           (STR2 .GT. 0)
      200  CONTINUE
      COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
      COND .AND. LV(STR3,L3)
                                                                ((LV(STR1,L1)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(STR3,L3)) .AND. (0 .LE.
                                                                STR1) .AND. (0 .LE. -STR1) .AND. (0
                                                                .LE. STR2) .AND. (0 .LT. STR2))
                                                                 .IMPLIES.
                                                                ((LV(STR1,L1)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(STR3,L3)) .AND. ((0 .NE.
                                                                -STR1) .OR. (0 .NE. -STR2)))

5
      COND LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3)
           SUBROUTINE CONCAT(STR1,STR2,STR3)
           COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
           INTEGER SPACE,STR1,STR2,STR3,TEMP1,FINDEN,POINTE,FAIL
           TEMP1=0
            .NOT. (STR1 .LT. 0)
            .NOT. (STR1 .GT. 0)
      500   .NOT. (STR2 .LT. 0)
            .NOT. (STR2 .GT. 0)
      COND STR1=0 .AND. STR2=0 .AND. LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3)
                                                                ((LV(STR1,L1)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(STR3,L3)) .AND. (0 .LE.
                                                                STR1) .AND. (0 .LE. -STR1) .AND. (0
                                                                .LE. STR2) .AND. (0 .LE. -STR2))
                                                                 .IMPLIES.
                                                                ((LV(STR1,L1)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(STR3,L3)) .AND. (0 .EQ.
                                                                STR2) .AND. (0 .EQ. STR1))

6
      COND STR1=0 .AND. STR2=0 .AND. LV(STR1,L1) .AND. LV(STR2,L2) .AND. LV(STR3,L3)
```

```
105   CALL COPY(STR1,STR3,0)
COND LV(STR1,L) .AND. LV(STR3,L3) .AND. STR1=STR3 .AND. STR2=0
```

```
((0 .EQ. STR1) .AND. (0 .EQ. STR2)
.AND. (LV(STR1,L1)) .AND.
(LV(STR2,L2)) .AND. (LV(STR3,L3)))
 .IMPLIES.
((LV(STR1,L1)) .AND. (LV(STR3,L3))
.AND. (0 .EQ. -STR1+STR3))
```

```
7
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
COND .AND. LV(STR3,L3)
      CALL COPY(TEMP1,STR3,0)
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
COND .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3
```

```
(((0 .NE. -STR1) .OR. (0 .NE.
-STR2)) .AND. (LV(STR1,L1)) .AND.
(LV(STR2,L2)) .AND. (LV(STR3,L3)))
 .IMPLIES.
((LV(STR1,L1)) .AND. (LV(STR2,L2))
.AND. (LV(STR3,L3)) .AND.
(LV(TEMP1,L3)) .AND. (0 .EQ.
STR3-TEMP1) .AND. ((0 .NE. -STR1)
.OR. (0 .NE. -STR2)))
```

```
8
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
COND .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3
      (STR1-STR2 .EQ. 0)
2     FINDEN=STP1+1
COND LV(STR1,L) .AND. LV(STR2,L2) .AND. LV(TEMP1,L3) .AND. LV(STR3,L3) .AND.
COND STR1=STR2 .AND. TEMP1=STR3 .AND. LIST.PIR(FINDEN-1,L)
```

```
(((0 .NE. -STR1) .OR. (0 .NE.
-STR2)) .AND. (LV(STR1,L1)) .AND.
(LV(STR2,L2)) .AND. (LV(STR3,L3))
.AND. (LV(TEMP1,L3)) .AND. (0 .EQ.
STR3-TEMP1) .AND. (0 .EQ. STP1-STR2))
 .IMPLIES.
((LV(STR1,L1)) .AND. (LV(STR2,L2))
.AND. (LV(TEMP1,L3)) .AND.
(LV(STR3,L3)) .AND.
(LIST.PIR(STR1,L1)) .AND. (0 .EQ.
-STR1+STR2))
```

```
9
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
COND .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3
     .NOT. (STR1-STR2 .EQ. 0)
COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
COND .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3 .AND. (STR1 .NE.
COND STR2)
```

```
(((0 .NE. -STR1) .OR. (0 .NE.
-STR2)) .AND. (LV(STR1,L1)) .AND.
```

```
                                                                    (LV(STR2,L2)) .AND. (LV(STR3,L3))
                                                                    .AND. (LV(TEMP1,L3)) .AND. (0 .EQ.
                                                                    STR3-TEMP1) .AND. (0 .NE. STR1-STR2))
                                                                     .IMPLIES.
                                                                    ((LV(STR1,L1)) .AND. (LV(STR2,L2))
                                                                    .AND. (LV(STR3,L3)) .AND.
                                                                    (LV(TEMP1,L3)) .AND. (0 .NE.
                                                                    -STR1+STR2) .AND. ((0 .NE. -STR1)
                                                                    .OR. (0 .NE. -STR2)))

10
      COND ((STR1 .NE. 0) .OR. (STR2 .NE. 0)) .AND. LV(STR1,L1) .AND. LV(STR2,L2)
      COND .AND. LV(STR3,L3) .AND. LV(TEMP1,L3) .AND. TEMP1=STR3 .AND. (STR1 .NE.
      COND STR2)
      1     CALL COPY(STR1,STR2,0)
      COND STR1=STR2 .AND. LV(STR1,L) .AND. LV(STR3,L3) .AND. LV(STR2,L) .AND.
      COND LV(TEMP1,L3) .AND. TEMP1=STR3
                                                                    (((0 .NE. -STR1) .OR. (0 .NE.
                                                                    -STR2)) .AND. (LV(STR1,L1)) .AND.
                                                                    (LV(STR2,L2)) .AND. (LV(STR3,L3))
                                                                    .AND. (LV(TEMP1,L3)) .AND. (0 .EQ.
                                                                    STR3-TEMP1) .AND. (0 .NE.
                                                                    -STR1+STR2))
                                                                     .IMPLIES.
                                                                    ((LV(STR1,L)) .AND. (LV(STR3,L3))
                                                                    .AND. (LV(STR2,L)) .AND.
                                                                    (LV(TEMP1,L3)) .AND. (0 .EQ.
                                                                    -STR1+STR2))

11
      COND STR1=STR2 .AND. LV(STR1,L) .AND. LV(STR3,L3) .AND. LV(STR2,L) .AND.
      COND LV(TEMP1,L3) .AND. TEMP1=STR3
             (STR1 .NE. 0)
      2      FINDEX=STR1+1
      COND LV(STR1,L) .AND. LV(STR2,L2) .AND. LV(TEMP1,L3) .AND. LV(STR3,L3) .AND.
      COND STR1=STR2 .AND. TEMP1=STR3 .AND. LIST.PIR(FINDEX-1,L)
                                                                    ((0 .EQ. -STR1+STR2) .AND.
                                                                    (LV(STR1,L)) .AND. (LV(STR3,L3))
                                                                    .AND. (LV(STR2,L)) .AND.
                                                                    (LV(TEMP1,L3)) .AND. (0 .EQ.
                                                                    STR3-TEMP1) .AND. (0 .NE. -STR1))
                                                                     .IMPLIES.
                                                                    ((LV(STR1,L)) .AND. (LV(STR2,L2))
                                                                    .AND. (LV(TEMP1,L3)) .AND.
                                                                    (LV(STR3,L3)) .AND.
                                                                    (LIST.PIR(STR1,L)))

12
      COND STR1=STR2 .AND. LV(STR1,L) .AND. LV(STR3,L3) .AND. LV(STR2,L) .AND.
      COND LV(TEMP1,L3) .AND. TEMP1=STR3
             .NOT. (STR1 .NE. 0)
      3      STR1=TEMP1
      COND LV(STR1,L) .AND. LV(STR3,L3) .AND. STR1=STR3 .AND. STR2=0
```

```
                                                                .AND. (LV(STR2,L)) .AND.
                                                                (LV(TEMP1,L3)) .AND. (0 .EQ.
                                                                STR3-TEMP1) .AND. (0 .EQ. -STR1))
                                                                 .IMPLIES.
                                                                ((LV(TEMP1,L)) .AND. (LV(STR3,L3))
                                                                .AND. (0 .EQ. STR2))

13
      COND LV(STR1,L) .AND. LV(STR2,L2) .AND. LV(TEMP1,L3) .AND. LV(STR3,L3) .AND.
      COND STR1=STR2 .AND. TEMP1=STR3 .AND. LIST.PIR(FINDEN-1,L)
      5     POINTE=SPACE(FINDEN)
            (POINTE .EQ. 0)
      5     SPACE(FINDEN)=TEMP1
      COND LV(STR1,LIST) .AND. LIST=COMP(L2,L3)
                                                                ((LV(STR1,L)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(TEMP1,L3)) .AND.
                                                                (LV(STR3,L3)) .AND. (0 .EQ.
                                                                -STR1+STR2) .AND. (0 .EQ.
                                                                STR3-TEMP1) .AND.
                                                                (LIST.PIR(-1+FINDEN,L)) .AND. (0
                                                                .EQ. SPACE(FINDEN)))
                                                                 .IMPLIES.
                                                                ((LV(STR1,LIST)) .AND. (0 .EQ.
                                                                COMP(L2,L3)-LIST))

14
      COND LV(STR1,L) .AND. LV(STR2,L2) .AND. LV(TEMP1,L3) .AND. LV(STR3,L3) .AND.
      COND STR1=STR2 .AND. TEMP1=STR3 .AND. LIST.PIR(FINDEN-1,L)
      5     POINTE=SPACE(FINDEN)
             .AND. (POINTE .EQ. 0)
      4     FINDEN=POINTE+1
      COND LV(STR1,L) .AND. LV(STR2,L2) .AND. LV(TEMP1,L3) .AND. LV(STR3,L3) .AND.
      COND STR1=STR2 .AND. TEMP1=STR3 .AND. LIST.PIR(FINDEN-1,L)
                                                                ((LV(STR1,L)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(TEMP1,L3)) .AND.
                                                                (LV(STR3,L3)) .AND. (0 .EQ.
                                                                -STR1+STR2) .AND. (0 .EQ.
                                                                STR3-TEMP1) .AND.
                                                                (LIST.PIR(-1+FINDEN,L)) .AND. (0
                                                                .NE. SPACE(FINDEN)))
                                                                 .IMPLIES.
                                                                ((LV(STR1,L)) .AND. (LV(STR2,L2))
                                                                .AND. (LV(TEMP1,L3)) .AND.
                                                                (LV(STR3,L3)) .AND.
                                                                (LIST.PIR(SPACE(FINDEN),L)))
```

2.5) CONCAT(STR1,STR2,STR3)

This subroutine creates a new string and sets STR1 to point to it. This new string is the concatenation of the two strings STR1 and STR2.

example: STR2 = "BIRD" STR3 = "DOG"
result : STR1 = "BIRDDOG"

If the original value of STR1 is not equal to STR2 and STR3, the string that STR1 was pointing to is first deleted i.e. returned to the string of available space.

The verification of CONCAT is as follows:

Path 1-5	The verification conditions are obvious.
Path 6&7	To verify the correctness of these two paths is to verify the correctness of return value of subroutine COPY(I,J,0).
Path 8	The hypothesis LV(STR1,L1).AND.LV(STR2,L2).AND. STR1=STR2 implies LV(STR1,L).AND. LIST.PIR(STR1,L) because L, L1 and L2 are equal as implied by hypothesis.
Path 9	The verification condition is obvious.
Path 10	Verification condition is same as Paths 6&7.
Path 11	Verification condition is same as Path 8.
Path 12	Verfication condition is obvious.
Path 13	The hypothesis LIST.PIR(-1+FINDEN,L).AND.(0.EQ. SPACE(FINDEN)).AND.LV(TEMP1,L3) implies SPACE(FINDEN-1) is last element of list L. Setting SPACE(FINDEN)=TEMP1 gives the specified conclusion.
Path 14	The verification condition is LIST.PIR(-1+FINDEN,L) .AND.(0.NE.SPACE(FINDEN)) implies LIST.PIR(SPACE(FINDEN),L).

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC    .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
COND   LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2)
      SUBROUTINE COPY(NAME,BEGIN,END)
      COMMON/BLOCK/SPACE(10000),FAIL,IFOO
      INTEGER SPACE,BEGIN,END,TEMP1,TEMP2,CHAR,POINTE,FAIL
C
C  THIS SUBROUTINE COPIES A STRING OR PORTION OF A STRING FROM THE
C  LOCATION BEGIN UP TO AND INCLUDING THE LOCATION END INTO A NEW STRING
C  CALLED NAME. IF BEGIN=0 OR END=0, IT ASSIGNS THE NULL STRING TO NAME.
C  (THE PARAMETERS BEGIN AND END ARE NOT CHANGED.)
C        AN ADDITIONAL FEATURE IS THAT A COMPLETE STRING CAN BE COPIED
C  BY SIMPLY PUTTING A #0# IN PLACE OF THE PARAMETER END. AN EXAMPLE IS,
C        TO SIMULATE THE SNOBOL STATEMENT STRING1 = STRING2
C        USE THE FORTRAN CALL COPY(STRING1,STRING2,0)
C
C  CHECK BEGIN FOR NULL
      CALL DELETB(NAME)
COND   LV(NAME,L1),L1=NIL
      IF(BEGIN) 3,1,3
    1 NAME=0
COND   LV(NAME,L1),L1=NIL
      RETURN
C  FIND A NEW CELL AND HAVE THE NEW STRING POINT TO IT
COND   LV(NAME,L1),L1=NIL
    3 NAME=NEWCEL(0)
COND   SPACE(NAME)=CAR(L),LV(FREE,CDR(L))
C  SAVE PARAMETERS
      TEMP1=NAME
      TEMP2=BEGIN
C  PULL OUT A CHARACTER FROM THE PORTION OF THE STRING THAT IS TO BE
C  COPIED AND PLACE IT IN THE NEW STRING
    4 CHAR=SPACE(TEMP2)
      SPACE(TEMP1)=CHAR
COND   SUBLIST(NAME,SPACE(TEMP1+1),L1),SUBLIST(BEGIN,SPACE(TEMP2+1),L2),
COND   SL1=SL2
C  DETERMINE IF ALL THE DESIRED CHARACTERS HAVE BEEN COPIED
      IF(END-TEMP2) 5,7,5
C  GET THE NEXT CHARACTER TO BE COPIED
    5 TEMP2=SPACE(TEMP2+1)
C  EXIT IF THE ZERO POINTER HAS BEEN FOUND
      IF(TEMP2) 6,7,6
C  FIND A NEW CELL
    6 POINTE=NEWCEL(0)
C  HAVE POINTER OF PREVIOUS CELL POINT TO THIS NEW CELL
      SPACE(TEMP1+1)=POINTE
C  REPEAT WHOLE COPY PROCESS USING THIS NEW CELL
      TEMP1=POINTE
      GO TO 4
C  INSERT ZERO POINTER AND RETURN
    7 SPACE(TEMP1+1)=0
COND   (LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2),L1=SL2)
COND         .OR. (LV(NAME,L1),LV(BEGIN,L2),L1=L2)
```

```
CC     LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC     SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC     .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
COND    LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2)
       SUBROUTINE COPY(NAME,BEGIN,END)
       COMMON/BLOCK/SPACE(10000),FAIL,IFOO
       INTEGER SPACE,BEGIN,END,TEMP1,TEMP2,CHAR,POINTE,FAIL
C
C  THIS SUBROUTINE COPIES A STRING OR PORTION OF A STRING FROM THE
C  LOCATION BEGIN UP TO AND INCLUDING THE LOCATION END INTO A NEW STRING
C  CALLED NAME. IF BEGIN=0 OR END=0, IT ASSIGNS THE NULL STRING TO NAME.
C  (THE PARAMETERS BEGIN AND END ARE NOT CHANGED.)
C         AN ADDITIONAL FEATURE IS THAT A COMPLETE STRING CAN BE COPIED
C  BY SIMPLY PUTTING A #0# IN PLACE OF THE PARAMETER END. AN EXAMPLE IS,
C         TO SIMULATE THE SNOBOL STATEMENT STRING1 = STRING2
C         USE THE FORTRAN CALL COPY(STRING1,STRING2,0)
C
C  CHECK BEGIN FOR NULL
       CALL DELTB(NAME)
COND    LV(NAME,L1),L1=NIL
       IF(BEGIN) 3,1,3
     1 NAME=0
COND    LV(NAME,L1),L1=NIL
       RETURN
C  FIND A NEW CELL AND HAVE THE NEW STRING POINT TO IT
COND    LV(NAME,L1),L1=NIL
     3 NAME=NEWCEL(0)
COND    SPACE(NAME)=CAR(L),LV(FREE,CDR(L))
C  SAVE PARAMETERS
       TEMP1=NAME
       TEMP2=BEGIN
C  PULL OUT A CHARACTER FROM THE PORTION OF THE STRING THAT IS TO BE
C  COPIED AND PLACE IT IN THE NEW STRING
     4 CHAR=SPACE(TEMP2)
       SPACE(TEMP1)=CHAR
COND    SUBLIST(NAME,SPACE(TEMP1+1),L1),SUBLIST(BEGIN,SPACE(TEMP2+1),L2),
COND    SL1=SL2
C  DETERMINE IF ALL THE DESIRED CHARACTERS HAVE BEEN COPIED
       IF(END-TEMP2) 5,7,5
C  GET THE NEXT CHARACTER TO BE COPIED
     5 TEMP2=SPACE(TEMP2+1)
C  EXIT IF THE ZERO POINTER HAS BEEN FOUND
       IF(TEMP2) 6,7,6
C  FIND A NEW CELL
     6 POINTE=NEWCEL(0)
C  HAVE POINTER OF PREVIOUS CELL POINT TO THIS NEW CELL
       SPACE(TEMP1+1)=POINTE
C  REPEAT WHOLE COPY PROCESS USING THIS NEW CELL
       TEMP1=POINTE
       GO TO 4
C  INSERT ZERO POINTER AND RETURN
     7 SPACE(TEMP1+1)=0
COND   (LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2),L1=SL2)
COND          .OR. (LV(NAME,L1),LV(BEGIN,L2),L1=L2)
```

```
RETURN
END
```

	PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
	COND LV(NAME,L1) .AND. SUBLIST(BEGIN,SPACE(END+1),L2)		
	SUBROUTINE COPY(NAME,BEGIN,END)		
	COMMON/BLOCK/SPACE(10000),FAIL,IFQQ		
	INTEGER SPACE,BEGIN,END,TEMP1,TEMP2,CHAR,POINTE,FAIL		
	CALL DELETB(NAME)		
	COND LV(NAME,L1) .AND. L1=NIL		
	IF(BEGIN .NE. 0)GO TO 3		
1	NAME=0	[NAME]	[]
	COND LV(NAME,L1) .AND. L1=NIL		
	RETURN		
	COND LV(NAME,L1) .AND. L1=NIL		
3	NAME=NEWCEL(0)	[NAME]	[NEWCEL]
	COND SPACE(NAME)=CAR(L) .AND. LV(FREE,CDR(L))		
	TEMP1=NAME	[TEMP1]	[NAME]
	TEMP2=BEGIN	[TEMP2]	[BEGIN]
4	CHAR=SPACE(TEMP2)	[CHAR]	[SPACE,TEMP2]
	SPACE(TEMP1)=CHAR	[SPACE]	[SPACE,TEMP1,CHAR]
	COND SUBLIST(NAME,SPACE(TEMP1+1),L1) .AND.		
	COND SUBLIST(BEGIN,SPACE(TEMP2+1),L2) .AND. SL1=SL2		
	IF(END-TEMP2 .EQ. 0)GO TO 7		
5	TEMP2=SPACE(TEMP2+1)	[TEMP2]	[SPACE,TEMP2]
	IF(TEMP2 .EQ. 0)GO TO 7		
6	POINTE=NEWCEL(0)	[POINTE]	[NEWCEL]
	SPACE(TEMP1+1)=POINTE	[SPACE]	[SPACE,TEMP1,POINTE]
	TEMP1=POINTE	[TEMP1]	[POINTE]
	GO TO 4		
7	SPACE(TEMP1+1)=0	[SPACE]	[SPACE,TEMP1]
	COND (LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2),L1 .EQ. SL2) .OR.		
	COND (LV(NAME,L1),LV(BEGIN,L2),L1 .EQ. L2)		
	RETURN		
	END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(NAME,L1) .AND. SUBLIST(BEGIN,SPACE(END+1),L2) SUBROUTINE COPY(NAME,BEGIN,END) COMMON/BLOCK/SPACE(10000),FAIL,IFOO INTEGER SPACE,BEGIN,END,TEMP1,TEMP2,CHAR,POINTE,FAIL CALL DELETB(NAME) COND LV(NAME,L1) .AND. L1=NIL	((LV(NAME,L1)) .AND. (SUBLIST(BEGIN,SPACE(1+END),L2))) .IMPLIES. ((LV(NAME,L1)) .AND. (0 .EQ. -L1+NIL))
2	COND LV(NAME,L1) .AND. L1=NIL (BEGIN .NE. 0) COND LV(NAME,L1) .AND. L1=NIL	((LV(NAME,L1)) .AND. (0 .EQ. -L1+NIL) .AND. (0 .NE. -BEGIN)) .IMPLIES. ((LV(NAME,L1)))
3	COND LV(NAME,L1) .AND. L1=NIL .NOT. (BEGIN .NE. 0) 1 NAME=0 COND LV(NAME,L1) .AND. L1=NIL	((LV(NAME,L1)) .AND. (0 .EQ. -L1+NIL) .AND. (0 .EQ. -BEGIN)) .IMPLIES. ((LV(0,L1)))
4	COND LV(NAME,L1) .AND. L1=NIL 3 NAME=NEWCEL(0) COND SPACE(NAME)=CAR(L) .AND. LV(FREE,CDR(L))	((LV(NAME,L1)) .AND. (0 .EQ. -L1+NIL)) .IMPLIES. ((LV(FREE,CDR(L))) .AND. (0 .EQ. CAR(L)-SPACE(NEWCEL(0))))
5	COND SPACE(NAME)=CAR(L) .AND. LV(FREE,CDR(L)) TEMP1=NAME TEMP2=BEGIN 4 CHAR=SPACE(TEMP2)	

```
        SPACE(TEMP1)=CHAR
COND SUBLIST(NAME,SPACE(TEMP1+1),L1) .AND. SUBLIST(BEGIN,SPACE(TEMP2+1),L2)
COND .AND. SL1=SL2
                                                    ((0 .EQ. CAR(L)-SPACE(NAME)) .AND.
                                                    (LV(FREE,CDR(L))))
                                                     .IMPLIES.
                                                    ((SUBLIST(NAME,SPACE(1+NAME),L1))
                                                    .AND.
                                                    (SUBLIST(BEGIN,SPACE(1+BEGIN),L2))
                                                    .AND. (0 .EQ. -SL1+SL2))

6
COND SUBLIST(NAME,SPACE(TEMP1+1),L1) .AND. SUBLIST(BEGIN,SPACE(TEMP2+1),L2)
COND .AND. SL1=SL2
       (END-TEMP2 .EQ. 0)
7      SPACE(TEMP1+1)=0
COND (LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2),L1 .EQ. SL2) .OR.
COND (LV(NAME,L1),LV(BEGIN,L2),L1 .EQ. L2)
                                                    ((SUBLIST(NAME,SPACE(1+TEMP1),L1))
                                                    .AND.
                                                    (SUBLIST(BEGIN,SPACE(1+TEMP2),L2))
                                                    .AND. (0 .EQ. -SL1+SL2) .AND. (0
                                                    .EQ. END-TEMP2))
                                                     .IMPLIES.
                                                    ((0 .EQ.
                                                    -LV(NAME,L1),SUBLIST(BEGIN,SPACE(1+EN
                                                    D),L2),L1+SL2) .OR. (0 .EQ.
                                                    L2-LV(NAME,L1),LV(BEGIN,L2),L1))

7
COND SUBLIST(NAME,SPACE(TEMP1+1),L1) .AND. SUBLIST(BEGIN,SPACE(TEMP2+1),L2)
COND .AND. SL1=SL2
        .NOT. (END-TEMP2 .EQ. 0)
5      TEMP2=SPACE(TEMP2+1)
       (TEMP2 .EQ. 0)
7      SPACE(TEMP1+1)=0
COND (LV(NAME,L1),SUBLIST(BEGIN,SPACE(END+1),L2),L1 .EQ. SL2) .OR.
COND (LV(NAME,L1),LV(BEGIN,L2),L1 .EQ. L2)
                                                    ((SUBLIST(NAME,SPACE(1+TEMP1),L1))
                                                    .AND.
                                                    (SUBLIST(BEGIN,SPACE(1+TEMP2),L2))
                                                    .AND. (0 .EQ. -SL1+SL2) .AND. (0
                                                    .NE. END-TEMP2) .AND. (0 .EQ.
                                                    SPACE(1+TEMP2)))
                                                     .IMPLIES.
                                                    ((0 .EQ.
                                                    -LV(NAME,L1),SUBLIST(BEGIN,SPACE(1+EN
                                                    D),L2),L1+SL2) .OR. (0 .EQ.
                                                    L2-LV(NAME,L1),LV(BEGIN,L2),L1))

8
COND SUBLIST(NAME,SPACE(TEMP1+1),L1) .AND. SUBLIST(BEGIN,SPACE(TEMP2+1),L2)
COND .AND. SL1=SL2
```

```
          .NOT. (END-TEMP2 .EQ. 0)                                                    1
5       TEMP2=SPACE(TEMP2+1)
          .NOT. (TEMP2 .EQ. 0)
6       POINTE=NEWCEL(0)
        SPACE(TEMP1+1)=POINTE
        TEMP1=POINTE
4       CHAR=SPACE(TEMP2)                                                           2
        SPACE(TEMP1)=CHAR
COND SUBLIST(NAME,SPACE(TEMP1+1),L1) .AND. SUBLIST(BEGIN,SPACE(TEMP2+1),L2)
COND .AND. SL1=SL2
                                                    ((SUBLIST(NAME,SPACE(1+TEMP1),L1))
                                                    .AND.
                                                    (SUBLIST(BEGIN,SPACE(1+TEMP2),L2))                  3
                                                    .AND. (0 .EQ. -SL1+SL2) .AND. (0
                                                    .NE. END-TEMP2) .AND. (0 .NE.
                                                    SPACE(1+TEMP2)))
                                                     .IMPLIES.
                                                    ((SUBLIST(NAME,SPACE(1+NEWCEL(0)),L1)
                                                    ) .AND.
                                                    (SUBLIST(BEGIN,SPACE(1+SPACE(1+TEMP2)
                                                    ),L2)))
```

2.6) COPY(NAME,BEGIN,END)

This subroutine creates a new string NAME whose characters are copied from pointer BEGIN to the pointer END INCLUSIVE. The string that NAME originally pointed to is first deleted.

VERIFICATION:

Path 1: Since subroutine DELETE(NAME) is called, I1=nil.

Path 2: obvious.

Path 3: Since I1=nil, hence IV(0,I1).

Path 4: Since subroutine calls were not implemented in the verifier, the call statement has not been considered. Here NEWCEL(0) creates a space for CAR(L).

Path 5: insufficient assertion.

Path 6: insufficient assertion.

Path 7: insuffieent assertion.

Path 8: insufficient assertion.

```
CC     LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), COR(L)).
CC     SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC     .AND. COR(SL)=SUBLIST(SPACE(J+1),K,L)))
COND    LV(FREE,L1),SUBLIST(J,K,L2)
        SUBROUTINE COPPY(I,J,K)
        COMMON/BLOCK/SPACE(10000),FAIL,FREE
        INTEGER SPACE,FAIL,FREE
        I=0
        IF(J.E).K)GO TO 9
        JJ=J
        KK=FREE
COND    SUBLIST(J,K,L2),(SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),
COND    (SL1=SL2)),LV(FREE,L1)
4       SPACE(KK)=SPACE(JJ)
        IF(SPACE(JJ+1).EQ.K)GO TO 3
        JJ=SPACE(JJ+1)
        IF(SPACE(KK+1).EQ.0)GO TO 2
        KK=SPACE(KK+1)
        IF(JJ.EQ.0)GO TO 5
        GO TO 4
3       I=FREE
        FREE=SPACE(KK+1)
        SPACE(KK+1)=0
COND    SUBLIST(I,FREE,L1),SUBLIST(J,K,L2),SPACE(KK)=SPACE(JJ),SL1=SL2,
COND    LV(I,L1),L1=SL2
        RETURN
2       WRITE(6,6)
6       FORMAT(30H OVERFLOW OF SPACE IN COPPY                )
COND    OVERFL)W(COPPY)
        STOP
5       WRITE(6,7)
7       FORMAT(50H ERROR IN PARAMETERS TO COPPY I=NULL AND RETURN        )
COND    ERROR(COPPY)
        RETURN
COND    I=0,J=K
9       RETURN
        END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(FREE,L1) .AND. SUBLIST(J,K,L2)		
SUBROUTINE COPPY(I,J,K)		
COMMON/BLOCK/SPACE(10000),FAIL,FREE		
INTEGER SPACE,FAIL,FREE		
I=0	[I]	[]
IF(J .EQ. K)GO TO 8		
JJ=J	[JJ]	[J]
KK=FREE	[KK]	[FREE]
COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1		
COND .EQ. SL2)) .AND. LV(FREE,L1)		
4 SPACE(KK)=SPACE(JJ)	[SPACE]	[SPACE,KK,JJ]
IF(SPACE(JJ+1) .EQ. K)GO TO 3		
JJ=SPACE(JJ+1)	[JJ]	[SPACE,JJ]
IF(SPACE(KK+1) .EQ. 0)GO TO 2		
KK=SPACE(KK+1)	[KK]	[SPACE,KK]
IF(JJ .EQ. 0)GO TO 5		
GO TO 4		
3 I=FREE	[I]	[FREE]
FREE=SPACE(KK+1)	[FREE]	[SPACE,KK]
SPACE(KK+1)=J	[SPACE]	[SPACE,KK]
COND SUBLIST(I,FREE,L1) .AND. SUBLIST(J,K,L2) .AND. SPACE(KK)=SPACE(JJ)		
COND .AND. SL1=SL2 .AND. LV(I,L1) .AND. L1=SL2		
RETURN		
2 WRITE(6,6)		
6 FORMAT(30H OVERFLOW OF SPACE IN COPPY)		
COND OVERFLOW(COPPY)		
STOP		
5 WRITE(6,7)		
7 FORMAT(50H ERROR IN PARAMETERS TO COPPY I=NULL AND RETURN)		
COND ERROR(COPPY)		
RETURN		
COND I=0 .AND. J=K		
8 RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(FREE,L1) .AND. SUBLIST(J,K,L2) SUBROUTINE COPY(I,J,K) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE I=0 (J .EQ. K) COND I=0 .AND. J=K	NONE
2	COND LV(FREE,L1) .AND. SUBLIST(J,K,L2) SUBROUTINE COPY(I,J,K) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE I=0 .NOT. (J .EQ. K) JJ=J KK=FREE COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1 .EQ. COND SL2)) .AND. LV(FREE,L1)	((LV(FREE,L1)) .AND. (SUBLIST(J,K,L2)) .AND. (0 .NE. -J+K)) .IMPLIES. ((SUBLIST(J,K,L2)) .AND. (LV(FREE,L1)) .AND. (SUBLIST(FREE,FREE,L1),SUBLIST(J,J,L2),0 .EQ. -SL1+SL2))
3	COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1 .EQ. COND SL2)) .AND. LV(FREE,L1) 4 SPACE(KK)=SPACE(JJ) (SPACE(JJ+1) .EQ. K) 3 I=FREE FREE=SPACE(KK+1) SPACE(KK+1)=0 COND SUBLIST(I,FREE,L1) .AND. SUBLIST(J,K,L2) .AND. SPACE(KK)=SPACE(JJ) .AND. COND SL1=SL2 .AND. LV(I,L1) .AND. L1=SL2	
2		((SUBLIST(J,K,L2)) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2) ,0 .EQ. -SL1+SL2) .AND. (LV(FREE,L1)) .AND. (0 .EQ. K-SPACE(1+JJ))) .IMPLIES. ((SUBLIST(FREE,SPACE(1+KK),L1)) .AND. (SUBLIST(J,K,L2)) .AND.
1		(LV(FREE,L1)) .AND. (0 .EQ. -L1+SL2)

```
                                                                .AND. (0 .EQ. -SL1+SL2))

4
      COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1 .EQ.
      COND SL2)) .AND. LV(FREE,L1)
      4     SPACE(KK)=SPACE(JJ)
             .NOT. (SPACE(JJ+1) .EQ. K)
            JJ=SPACE(JJ+1)
            (SPACE(KK+1) .EQ. 0)
      2      WRITE(6,6)
      6      FORMAT(30H OVERFLOW OF SPACE IN COPPY          )
      COND (OVERFLOW(COPPY)
                                                                ((SUBLIST(J,K,L2)) .AND.
                                                                (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2)
                                                                ,0 .EQ. -SL1+SL2) .AND.
                                                                (LV(FREE,L1)) .AND. (0 .NE.
                                                                K-SPACE(1+JJ)) .AND. (0 .EQ.
                                                                SPACE(1+KK)))
                                                                 .IMPLIES.
                                                                ((OVERFLOW(COPPY)))

5
      COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1 .EQ.
      COND SL2)) .AND. LV(FREE,L1)
      4     SPACE(KK)=SPACE(JJ)
             .NOT. (SPACE(JJ+1) .EQ. K)
            JJ=SPACE(JJ+1)
             .NOT. (SPACE(KK+1) .EQ. 0)
            KK=SPACE(KK+1)
            (JJ .EQ. 0)
      5     WRITE(6,7)
      7     FORMAT(50H ERROR IN PARAMETERS TO COPPY I=NULL AND RETURN       )
      COND ERROR(COPPY)
                                                                ((SUBLIST(J,K,L2)) .AND.
                                                                (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2)
                                                                ,0 .EQ. -SL1+SL2) .AND.
                                                                (LV(FREE,L1)) .AND. (0 .NE.
                                                                K-SPACE(1+JJ)) .AND. (0 .NE.
                                                                SPACE(1+KK)) .AND. (0 .EQ.
                                                                SPACE(1+JJ)))
                                                                 .IMPLIES.
                                                                ((ERROR(COPPY)))

6
      COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1 .EQ.
      COND SL2)) .AND. LV(FREE,L1)
      4     SPACE(KK)=SPACE(JJ)
             .NOT. (SPACE(JJ+1) .EQ. K)
            JJ=SPACE(JJ+1)
             .NOT. (SPACE(KK+1) .EQ. 0)
            KK=SPACE(KK+1)
             .NOT. (JJ .EQ. 0)
      COND SUBLIST(J,K,L2) .AND. (SUBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2),(SL1 .EQ.
```

```
      G(ND SL2)) .AND. LV(FREE,L1)
                                                  ((SUBLIST(J,K,L2)) .AND.
                                                  (SJBLIST(FREE,KK,L1),SUBLIST(J,JJ,L2)
                                                  ,0 .EQ. -SL1+SL2) .AND.
                                                  (LV(FREE,L1)) .AND. (0 .NE.
                                                  K-SPACE(1+JJ)) .AND. (0 .NE.
                                                  SPACE(1+KK)) .AND. (0 .NE.
                                                  SPACE(1+JJ)))
                                                   .IMPLIES.
                                                  ((SUBLIST(J,K,L2)) .AND.
                                                  (LV(FREE,L1)) .AND.
                                                  (SUBLIST(FREE,SPACE(1+KK),L1),SUBLIST
                                                  (J,SPACE(1+JJ),L2),0 .EQ. -SL1+SL2))
```

2.7) COPY(I,J,K)

This is the same as COPY except it copies up to but not including the second pointer. When K=0, the string is copied all the way to the end. Unlike COPY, however, the string that I originally points to is not deleted.

VERIFICATION:

Path 1: proved.

Path 2: obvious.

Path 3: insufficient assertion.

Path 4: Since output statement has not been implemented into the verifier, assertions are not enough to verify this path.

Path 5: same as path 4.

Path 6: insufficient assertion.

```
CC   IALPHN(27)=1H1, IALPHN(28)=1H1, IALPHN(29)=1H2, IALPHN(30)=1H3,
CC   IALPHN(31)=1H4, IALPHN(32)=1H5, IALPHN(33)=1H6, IALPHN(34)=1H7,
CC   IALPHN(35)=1H8, IALPHN(36)=1H9
CC   ALL(F(X),X,I,J) .IN. (F(X), X.IN.(I..J))
CC   TYPE(L,1)=*REAL*, TYPE(L,2)=*INTEGER*, TYPE(L,3)=*STRING*
CC   EXISTS(F(X),X,I,J) .IN. (F(X),I.LE.X,X.LE.J)
CC   LIST.PIR(K,L) .IN. EXISTS(K=ADDR(I,L),I,1,LENGTH(L))
C    LV(LP,L) MEANS THE LIST VALUE OF LP IS L
CC   LV(LP,L) = (K(1..N)), ((I).LE.K(I), K(I).LE.5000,
CC   SPACE(2*K(I)-1) = L(I), SPACE(2*K(I)) = 2*K(I+1) -1),
CC   I .IN. (1..N)), LP = 2*K(1) - 1)
C
COND   LV(STRING,L), TYPE(L,1)
       INTEGER FUNCTION DECODE(STRING)
       COMMON/BLOCK/SPACE(10000),FAIL,IPOO
       COMMON/ALPHN/IALPHN(68)
       INTEGER SPACE,FAIL,STRING
C  THIS FUNCTION CONVERTS A STRING INTO A NUMBER
       INPUT=STRING
COND   LV(STRING,L), LIST.PIR(INPUT,L), INPUT=STRING
       DECODE=0
     1 ITEMP=SPACE(INPUT)
       I = 27
COND   LIST.PIR(INPUT,L),((DD=DECODE).OR.((DD=0).AND.(INPUT=STRING))),
COND   ALL((SPACE(INPUT).NE.IALPHN(J)),J,27,(I-1)),(ITEMP=SPACE(INPUT)),
COND   (I.LE.36)
6      IF(ITEMP.EQ.IALPHN(I))GO TO 4
       I = I + 1
       IF (I .LE. 36) GO TO 6
       GO TO 5
4      DECODE=I-27+DECODE
COND   LIST.PIR(INPUT,L),(DECODE=-27+DD+1),EXISTS((SPACE(INPUT)=
COND   IALPHN(J)),J,27,36)
       INPUT=SPACE(INPUT+1)
       IF(INPUT) 2,3,2
     2 DECODE=DECODE*10
       GO TO 1
5      WRITE(6,8)
8      FORMAT(1LX,40H***** INPUT STRING IS NOT A NUMBER *****)
COND   (EXISTS((DECODE=-27+DD+I),I,27,36)).OR.((DECODE=00),(I.GT.36))
     3 RETURN
       END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(STRING,L) .AND. TYPE(L,1)		
INTEGER FUNCTION DECODE(STRING)		
COMMON/BLOCK/SPACE(10000),FAIL,IFOO		
COMMON/ALPHNU/IALPHN(63)		
INTEGER SPACE,FAIL,STRING		
INPUT=STRING	(INPUT)	(STRING)
COND LV(STRING,L) .AND. LIST.PIR(INPUT,L) .AND. INPUT=STRING		
DECODE=0	(DECODE)	()
1 ITEMP=SPACE(INPUT)	(ITEMP)	(SPACE,INPUT)
I=27	(I)	()
COND LIST.PIR(INPUT,L) .AND. ((IOO .EQ. DECODE) .OR. ((IOO .EQ. 0) .AND.		
COND (INPUT .EQ. STRING))) .AND. ALL((SPACE(INPUT) .NE.		
COND IALPHN(J)),J,27,(I-1)) .AND. (ITEMP .EQ. SPACE(INPUT))' .AND. (I		
COND .LE. 36)		
6 IF(ITEMP .EQ. IALPHN(I))GO TO 4		
I=I+1	(I)	(I)
IF(I .LE. 36)GO TO 6		
GO TO 5		
4 IDCODE=I-27+DECODE	(DECODE)	(I,DECODE)
COND LIST.PIR(INPUT,L) .AND. (DECODE .EQ. -27+DO+I) .AND.		
COND EXISTS((SPACE(INPUT) .EQ. IALPHN(J)),J,27,36)		
INPUT=SPACE(INPUT+1)	(INPUT)	(SPACE,INPUT)
IF(INPUT .EQ. 0)GO TO 3		
2 DECODE=DECOD.*10	(DECODE)	(DECODE)
GO TO 1		
5 WRITE(6,8)		
8 FORMAT(10X,40H(I* INPUT STRING IS NOT A NUMBER UU*)		
COND (EXISTS((DECODE .EQ. -27+DO+I),I,27,36)) .OR. ((DECODE .EQ. DO),(I		
COND .GT. 36))		
3 RETURN		
END		

```
                                                                    .EQ.
                                                                    IALPHN(J)-SPACE(INPUT),J,27,3C))
                                                                    .AND. (0 .EQ. DO-DECODE))

4
      COND (LIST.PIR(INPUT,L) .AND. ((DO) .EQ. DECODE) .OR. ((DO .EQ. 0) .AND. (INPUT
      COND .EQ. STRING))) .AND. ALL((SPACE(INPUT) .NE. IALPHN(J)),J,27,(I-1)) .AND.
      COND (ITEMP .EQ. SPACE(INPUT)) .AND. (I .LE. 36)
      6       .NOT. (ITEMP .EQ. IALPHN(I))
              I=I+1
              (I .LE. 36)
      COND LIST.PIR(INPUT,L) .AND. ((DO .EQ. DECODE) .OR. ((DO .EQ. 0) .AND. (INPUT
      COND .EQ. STRING))) .AND. ALL((SPACE(INPUT) .NE. IALPHN(J)),J,27,(I-1)) .AND.
      COND (ITEMP .EQ. SPACE(INPUT)) .AND. (I .LE. 36)
                                                                    ((LIST.PIR(INPUT,L)) .AND. ((0 .EQ.
                                                                    -DO+DECODE) .OR. ((0 .EQ. DO) .AND.
                                                                    (0 .EQ. -INPUT+STRING))) .AND.
                                                                    (ALL(0 .NE.
                                                                    IALPHN(J)-SPACE(INPUT),J,27,-1+I))
                                                                    .AND. (0 .EQ. -ITEMP+SPACE(INPUT))
                                                                    .AND. (0 .LE. 36-I) .AND. (0 .NE.
                                                                    IALPHN(I)-ITEMP) .AND. (0 .LE. 35-I))
                                                                     .IMPLIES.
                                                                    ((LIST.PIR(INPUT,L)) .AND. (ALL(0
                                                                    .NE. IALPHN(J)-SPACE(INPUT),J,27,1))
                                                                    .AND. ((0 .EQ. -DO+DECODE) .OR. ((0
                                                                    .EQ. DO) .AND. (0 .EQ.
                                                                    -INPUT+STRING))))

5
      COND LIST.PIR(INPUT,L) .AND. ((D) .EQ. DECODE) .OR. ((DO .EQ. 0) .AND. (INPUT
      COND .EQ. STRING))) .AND. ALL((SPACE(INPUT) .NE. IALPHN(J)),J,27,(I-1)) .AND.
      COND (ITEMP .EQ. SPACE(INPUT)) .AND. (I .LE. 36)
      6       .NOT. (ITEMP .EQ. IALPHN(I))
              I=I+1
              .NOT. (I .LE. 36)
      5       WRITE(6,8)
      8       FORMAT(10X,40HUU* INPUT STRING IS NOT A NUMBER UU*)
      COND (EXISTS((DECODE .EQ. -27+DO+I),I,27,36)) .OR. ((DECODE .EQ. DO),(I .GT.
      COND 36))
                                                                    ((LIST.PIR(INPUT,L)) .AND. ((0 .EQ.
                                                                    -DO+DECODE) .OR. ((0 .EQ. DO) .AND.
                                                                    (0 .EQ. -INPUT+STRING))) .AND.
                                                                    (ALL(0 .NE.
                                                                    IALPHN(J)-SPACE(INPUT),J,27,-1+I))
                                                                    .AND. (0 .EQ. -ITEMP+SPACE(INPUT))
                                                                    .AND. (0 .LE. 36-I) .AND. (0 .NE.
                                                                    IALPHN(I)-ITEMP) .AND. (0 .LT.
                                                                    -35+I))
                                                                     .IMPLIES.
                                                                    ((EXISTS(0 .EQ.
                                                                    -26+DO-DECODE+I,1+I,27,36)) .OR. (0
                                                                    .EQ. DO-DECODE,0 .LT. -35+I))
```

```
6
COND LIST.PIR(INPUT,L) .AND. (DECODE .EQ. -27+D0+I) .AND. EXISTS((SPACE(INPUT)
COND .EQ. IALPH(J)),J,27,36)
      INPUT=SPACE(INPUT+1)
      (INPUT .EQ. 0)
COND (EXISTS((DECODE .EQ. -27+D0+I),I,27,36)) .OR. ((DECODE .EQ. D0),(I .GT.
COND 36))
```

```
((LIST.PIR(INPUT,L)) .AND. (0 .EQ.
-27+D0-DECODE+I) .AND. (EXISTS(0
.EQ.
IALPHN(J)-SPACE(INPUT),J,27,36))
.AND. (0 .EQ. SPACE(1+INPUT)))
 .IMPLIES.
((EXISTS(0 .EQ.
-27+D0-DECODE+I,I,27,36)) .OR. (0
.EQ. D0-DECODE,0 .LT. -36+I))
```

```
1
COND LIST.PIR(INPUT,L) .AND. (DECODE .EQ. -27+D0+I) .AND. EXISTS((SPACE(INPUT)
COND .EQ. IALPH(J)),J,27,36)
      INPUT=SPACE(INPUT+1)
       .NOT. (INPUT .EQ. 0)
2     DECODE=DECODE*10
1     ITEMP=SPACE(INPUT)
      I=27
COND LIST.PIR(INPUT,L) .AND. ((D0) .EQ. DECODE) .OR. ((D0 .EQ. 0) .AND. (INPUT
COND .EQ. STRING)) .AND. ALL((SPACE(INPUT) .NE. IALPHN(J)),J,27,(I-1)) .AND.
COND ((ITEMP .EQ. SPACE(INPUT)) .AND. (I .LE. 36)
```

```
((LIST.PIR(INPUT,L)) .AND. (0 .EQ.
-27+D0-DECODE+I) .AND. (EXISTS(0
.EQ.
IALPHN(J)-SPACE(INPUT),J,27,36))
.AND. (0 .NE. SPACE(1+INPUT)))
 .IMPLIES.
((LIST.PIR(SPACE(1+INPUT),L)) .AND.
(ALL(0 .NE.
IALPHN(J)-SPACE(SPACE(1+INPUT)),J,27,
26)) .AND. ((0 .EQ. -D0+10*DECODE)
.OR. ((0 .EQ. D0) .AND. (0 .EQ.
-SPACE(1+INPUT)+STRING))))
```

2.8) DECODE(STRING)

STRING is a string of numerals and DECODE returns the numerical value that STRING represents.

example: STRING = "732"
result : DECODE(STRING) = 732

The verification of this function subprogram requires setting DO as initial value of DECODE in the initial assertion of each path. The verification conditions of DECODE routine are as follows:

Path 1 The verification condition is obvious.

Path 2 Since ALL(0.NE.IALPHN(J)-SPACE(INPUT),J,27,26)= .TRUE., the resulting simplified form of conclusion is ((LIST.PIR(INPUT,L)).AND.((0.EQ.DO).OR.((0.EQ.DO) .AND.(0.EQ.-INPUT+STRING)))), and the verification is obviousis obvious.

Path 3 There are two verification conditions of this path:

(i) ((DO.EQ.DECODE).OR.((DO.EQ.0).AND.(INPUT.EQ. STRING))) implies (0.EQ.DO-DECODE);

(ii) the remaining statements of the hypothesis and conclusion not listed in (i).

Path 4 In short, the property that (ALL(F(X),X,I,J).AND. F(X+1)) implies ALL(F(X),X,I,J+1)) is needed in the verification of this path. The verification condition is then

(ALL(0.NE.IALPHN(J)-SPACE(INPUT),J,27,-I±1).AND. (0.EQ.-ITEMP+SPACE(INPUT)).AND.(0.LE.36-I).AND. (0.NE.IALPHN(I)-ITEMP))
implies
ALL(0.NE.IALPHN(J)-SPACE(INPUT),J,27,I).

Path 5 The verification conditions of this path can be

shown in two seperate steps:

(i) ((0.LE.36-I).AND.(0.LT.-35+I)) implies I=36. Therefore, EXISTS(0.EQ.-26+DO-DECODE+I,I+1, 27,36)=.TRUE.;

(ii) the remaining verification condition is ((0.EQ.-DO+DECODE).OR.((0.EQ.DO).AND.(0.EQ. -INPUT+STRING))).AND.(0.LT.-35+I) implies (0.EQ.DO-DECODE, 0.LT.-35+I).

Path 6 The verification condition of this path is (0.EQ.-27+DO-DECODE+I) implies (EXISTS(0.EQ.-27+DO-DECODE+I,I,27,36).OR.(0.EQ. DO-DECODE, 0.LT.-36+I)).

Path 7 Since ALL(0.NE.IALPHN(J)-SPACE(SPACE(1+INPUT)),J, 27,26)=.TRUE., the verification condition is simplified to (LIST.PIR(INPUT,L).AND.(0.NE.SPACE(1+INPUT)) implies (LIST.PIR(SPACE(1+INPUT),L).AND.(0.EQ.-DO+10*DECODE)). The value of DO is reset in the conclusion and do not require further verification.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
COND   LV(STRING,L1),ELEM .GE. 0
      SUBROUTINE DELETA(STRING,ELEM)
CC SPACE(N)
      COMMON/BLOCK/SPACE(10000),F,FREE
CC SPACE(N)
      INTEGER SPACE,F,FREE,STRING,ELEM,TEMP
C WALK THRU LIST TO ELEM-1 ELEMENT
      INDEX = STRING
      IF(ELEM -1) 290,280,200
200   K = ELEM - 1
      J=1
COND   SPACE(INDEX) .NE. ELT(ELEM,L1),SPACE(INDEX)=ELT(J,L1),
COND   K=ELEM-1,J .LE. K
  270 INDEX=SPACE(INDEX+1)
      J=J+1
      IF(J .LE. K) GO TO 270
COND   SPACE(INDEX)=ELT(ELEM,L1)
C DELETE CELL
280   TEMP = SPACE(INDEX+1)
      SPACE(INDEX+1) = SPACE(TEMP+1)
C ADD CELL TO FREE LIST
      SPACE(TEMP+1) = FREE
      FREE = TEMP
COND   SPACE(FREE)=ELT(ELEM+1,L1)
      RETURN
C  DELETE FIRST ELEMENT OF A STRING
290   STRING = SPACE(INDEX + 1)
      SPACE(INDEX + 1) = FREE
      FREE = INDEX
COND   SPACE(FREE)=ELT(ELEM+1,L1)
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(STRING,L1) .AND. ELEM .GE. 0		
SUBROUTINE DELETA(STRING,ELEM)		
COMMON/BLOCK/SPACE(10000),F,FREE		
INTEGER SPACE,F,FREE,STRING,ELEM,TEMP		
INDEX=STRING	[INDEX]	[STRING]
IF(ELEM-1 .LT. 0)GO TO 290		
IF(ELEM-1 .EQ. 0)GO TO 280		
200 K=ELEM-1	[K]	[ELEM]
J=1	[J]	[ ]
COND SPACE(INDEX) .NE. ELT(ELEM,L1) .AND. SPACE(INDEX)=ELT(J,L1) .AND.		
COND K=ELEM-1 .AND. J .LE. K		
270 INDEX=SPACE(INDEX+1)	[INDEX]	[SPACE,INDEX]
J=J+1	[J]	[J]
IF(J .LE. K)GO TO 270		
COND SPACE(INDEX)=ELT(ELEM,L1)		
280 TEMP=SPACE(INDEX+1)	[TEMP]	[SPACE,INDEX]
SPACE(INDEX+1)=SPACE(TEMP+1)	[SPACE]	[SPACE,INDEX,TEMP]
SPACE(TEMP+1)=FREE	[SPACE]	[SPACE,TEMP,FREE]
FREE=TEMP	[FREE]	[TEMP]
COND SPACE(FREE)=ELT(ELEM+1,L1)		
RETURN		
290. STRING=SPACE(INDEX+1)	[STRING]	[SPACE,INDEX]
SPACE(INDEX+1)=FREE	[SPACE]	[SPACE,INDEX,FREE]
FREE=INDEX	[FREE]	[INDEX]
COND SPACE(FREE)=ELT(ELEM+1,L1)		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1'	COND LV(STRING,L1) .AND. ELEM .GE. 0 SUBROUTINE DELETA(STRING,ELEM) COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,F,FREE,STRING,ELEM,TEMP INDEX=STRING (ELEM-1 .LT. 0) 290 STRING=SPACE(INDEX+1) SPACE(INDEX+1)=FREE FREE=INDEX COND SPACE(FREE)=ELT(ELEM+1,L1)	((LV(STRING,L1)) .AND. (0 .LE. ELEM) .AND. (0 .LT. 1-ELEM)) .IMPLIES. ((0 .EQ. ELT(1+ELEM,L1)-SPACE(STRING)))
2	COND LV(STRING,L1) .AND. ELEM .GE. 0 SUBROUTINE DELETA(STRING,ELEM) COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,F,FREE,STRING,ELEM,TEMP INDEX=STRING .NOT. (ELEM-1 .LT. 0) (ELEM-1 .EQ. 0) COND SPACE(INDEX)=ELT(ELEM,L1)	((LV(STRING,L1)) .AND. (0 .LE. ELEM) .AND. (0 .LE. -1+ELEM) .AND. (0 .EQ. -1+ELEM)) .IMPLIES. ((0 .EQ. ELT(ELEM,L1)-SPACE(STRING)))
3	COND LV(STRING,L1) .AND. ELEM .GE. 0 SUBROUTINE DELETA(STRING,ELEM) COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,F,FREE,STRING,ELEM,TEMP INDEX=STRING .NOT. (ELEM-1 .LT. 0) .NOT. (ELEM-1 .EQ. 0) 200 K=ELEM-1 J=1 COND SPACE(INDEX) .NE. ELT(ELEM,L1) .AND. SPACE(INDEX)=ELT(J,L1) .AND. K=ELEM-1 COND .AND. J .LE. K	((LV(STRING,L1)) .AND. (0 .LE. ELEM) .AND. (0 .LE. -1+ELEM) .AND. (0 .NE. -1+ELEM)) .IMPLIES. ((0 .LE. -2+ELEM) .AND. (0 .EQ.

```
                                                                    ELT(1,L1)-SPACE(STRING)) .AND. (0
                                                                    .NE. ELT(ELEM,L1)-SPACE(STRING)))

4
      COND SPACE(INDEX) .NE. ELT(ELEM,L1) .AND. SPACE(INDEX)=ELT(J,L1) .AND. K=ELEM-1
      COND .AND. J .LE. K
      270   INDEX=SPACE(INDEX+1)
            J=J+1
            (J .LE. K)
      COND SPACE(INDEX) .NE. ELT(ELEM,L1) .AND. SPACE(INDEX)=ELT(J,L1) .AND. K=ELEM-1
      COND .AND. J .LE. K
                                                                    ((0 .NE. ELT(ELEM,L1)-SPACE(INDEX))
                                                                    .AND. (0 .EQ.
                                                                    ELT(J,L1)-SPACE(INDEX)) .AND. (0
                                                                    .EQ. -1+ELEM-K) .AND. (0 .LE. -J+K)
                                                                    .AND. (0 .LE. -1-J+K))
                                                                     .IMPLIES.
                                                                    ((0 .EQ.
                                                                    ELT(1+J,L1)-SPACE(SPACE(1+INDEX)))
                                                                    .AND. (0 .NE.
                                                                    ELT(ELEM,L1)-SPACE(SPACE(1+INDEX))))

5
      COND SPACE(INDEX) .NE. ELT(ELEM,L1) .AND. SPACE(INDEX)=ELT(J,L1) .AND. K=ELEM-1
      COND .AND. J .LE. K
      270   INDEX=SPACE(INDEX+1)
            J=J+1
             .NGT. (J .LE. K)
      COND SPACE(INDEX)=ELT(ELEM,L1)
                                                                    ((0 .NE. ELT(ELEM,L1)-SPACE(INDEX))
                                                                    .AND. (0 .EQ.
                                                                    ELT(J,L1)-SPACE(INDEX)) .AND. (0
                                                                    .EQ. -1+ELEM-K) .AND. (0 .LE. -J+K)
                                                                    .AND. (0 .LT. 1+J-K))
                                                                     .IMPLIES.
                                                                    ((0 .EQ.
                                                                    ELT(ELEM,L1)-SPACE(SPACE(1+INDEX))))

6
      COND SPACE(INDEX)=ELT(ELEM,L1)
      280   TEMP=SPACE(INDEX+1)
            SPACE(INDEX+1)=SPACE(TEMP+1)
            SPACE(TEMP+1)=FREE
            FREE=TEMP
      COND SPACE(FREE)=ELT(ELEM+1,L1)
                                                                    ((0 .EQ. ELT(ELEM,L1)-SPACE(INDEX)))
                                                                     .IMPLIES.
                                                                    ((0 .EQ.
                                                                    ELT(1+ELEM,L1)-SPACE(SPACE(1+INDEX))))
                                                                    )
```

2.9) DELETA(STRING,ELEM)

DELETA deletes from the string STRING, the (ELEM+1) element and returns to the string of available space.

example: STRING = "DATA" ELEM=0
result : STRING = "ATA" and the length of the string of available space is increased by 1.

VERIFICATION:

PATH 1: Since ELEM=0, SPACE(STRING)=ELT(1,L1).

PATH 2: Since ELEM=1, SPACE(STRING)=ELT(1,L1).

PATH 3: Since ELEM .GE. 2, SPACE(STRING)≠ELT(ELEM,L1).

PATH 4: Since ELT(J,L1)=SPACE(INDEX), hence
ELT(J+1,L1)=SPACE(SPACE(INDEX+1)).
Since ELEM≠J+1, hence
ELT(ELEM,L1)≠SPACE(SPACE(INDEX+1)).

PATH 5: Since ELEM=J+1 in this path, hence
ELT(ELEM,L1)=SPACE(SPACE(INDEX+1)).

PATH 6: Obvious.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    CDRN(0,L)=L.
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
COND   LV(LINE,L1)
      SUBROUTINE DELETB(LINE)
      COMMON/BLOCK/SPACE(10000),FAIL,FREE
      INTEGER SPACE,FAIL,FREE
      IF(LINE)6,2,1
1       I=LINE
5     IF(SPACE(I+1))3,4,3
COND   SPACE(I) .NE. ELT(LENGTH(L1),L1)
3     I=SPACE(I+1)
      GO TO 5
COND   SPACE(I)=ELT(LENGTH(L1),L1)
4     SPACE(I+1)=FREE
      FREE=LINE
      LINE=0
COND   LV(LINE,L1),L1=NIL
2     RETURN
6      PRINT66
      KAPAN=SPACE(IZOJP)
66     FORMAT(/1X30HERROR IN ARG TO DELETB                    )
COND   ERROR(DELETB)
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(LINE,L1)		
SUBROUTINE DELETB(LINE)		
COMMON/BLOCK/SPACE(10000),FAIL,FREE		
INTEGER SPACE,FAIL,FREE		
IF(LINE .LT. 0)GO TO 6		
IF(LINE .EQ. 0)GO TO 2		
1 I=LINE	[I]	[LINE]
5 IF(SPACE(I+1) .EQ. 0)GO TO 4		
COND SPACE(I) .NE. ELT(LENGTH(L1),L1)		
3 I=SPACE(I+1)	[I]	[SPACE,I]
GO TO 5		
COND SPACE(I)=ELT(LENGTH(L1),L1)		
4 SPACE(I+1)=FREE	[SPACE]	[SPACE,I,FREE]
FREE=LINE	[FREE]	[LINE]
LINE=0	[LINE]	[]
COND LV(LINE,L1) .AND. L1=NIL		
2 RETURN		
6 PRINT66		
KAPAN=SPACE(IZOOP)	[KAPAN]	[SPACE,IZOOP]
66 FORMAT(/1X30HERROR IN ARG TO DELETB		
COND ERROR(DELETB)		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE (LINE .LT. 0) 6 PRINT66 KAPAN=SPACE(1ZOOP) 66 FORMAT(/1X30HERROR IN ARG TO DELETB COND ERROR(DELETB)	((LV(LINE,L1)) .AND. (0 .LT. -LINE)) .IMPLIES. ((ERROR(DELETB)))
2	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE .NOT. (LINE .LT. 0) (LINE .EQ. 0) COND LV(LINE,L1) .AND. L1=NIL	((LV(LINE,L1)) .AND. (0 .LT. LINE) .AND. (0 .EQ. LINE)) .IMPLIES. ((LV(LINE,L1)) .AND. (0 .EQ. -L1+NIL))
3	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE .NOT. (LINE .LT. 0) .NOT. (LINE .EQ. 0) 1 I=LINE 5 (SPACE(I+1) .EQ. 0) COND SPACE(I)=ELT(LENGTH(L1),L1)	((LV(LINE,L1)) .AND. (0 .LT. LINE) .AND. (0 .NE. LINE) .AND. (0 .EQ. SPACE(1+LINE))) .IMPLIES. ((0 .EQ. ELT(LENGTH(L1),L1)-SPACE(LINE)))
4	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE .NOT. (LINE .LT. 0)	

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE (LINE .LT. 0) 6 PRINT66 KAPAN=SPACE(1ZOOP) 66 FORMAT(/1X30HERROR IN ARG TO DELETB COND ERROR(DELETB)	((LV(LINE,L1)) .AND. (0 .LT. -LINE)) .IMPLIES. ((ERROR(DELETB)))
2	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE .NOT. (LINE .LT. 0) (LINE .EQ. 0) COND LV(LINE,L1) .AND. L1=NIL	((LV(LINE,L1)) .AND. (0 .LT. LINE) .AND. (0 .EQ. LINE)) .IMPLIES. ((LV(LINE,L1)) .AND. (0 .EQ. -L1+NIL))
3	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE .NOT. (LINE .LT. 0) .NOT. (LINE .EQ. 0) 1 I=LINE 5 (SPACE(I+1) .EQ. 0) COND SPACE(I)=ELT(LENGTH(L1),L1)	((LV(LINE,L1)) .AND. (0 .LT. LINE) .AND. (0 .NE. LINE) .AND. (0 .EQ. SPACE(1+LINE))) .IMPLIES. ((0 .EQ. ELT(LENGTH(L1),L1)-SPACE(LINE)))
4	COND LV(LINE,L1) SUBROUTINE DELETB(LINE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE .NOT. (LINE .LT. 0)	

```
        .NOT. (LINE .EQ. 0)                                                    1
1     I=LINE
5      .NOT. (SPACE(I+1) .EQ. 0)
COND SPACE(I) .NE. ELT(LENGTH(L1),L1)
                                                  ((LV(LINE,L1)) .AND. (0 .LE. LINE) .AND. (0
                                                  .NE. LINE) .AND. (0 .NE. SPACE(1+LINE)))
                                                   .IMPLIES.                    2
                                                  ((0 .NE. ELT(LENGTH(L1),L1)-SPACE(LINE)))

5
COND SPACE(I) .NE. ELT(LENGTH(L1),L1)
3     I=SPACE(I+1)                                                             3
5     (SPACE(I+1) .EQ. 0)
COND SPACE(I)=ELT(LENGTH(L1),L1)
                                                  ((0 .NE. ELT(LENGTH(L1),L1)-SPACE(I)) .AND.
                                                  (0 .EQ. SPACE(1+SPACE(1+I))))
                                                   .IMPLIES.
                                                  ((0 .EQ.
                                                  ELT(LENGTH(L1),L1)-SPACE(SPACE(1+I))))

6
COND SPACE(I) .NE. ELT(LENGTH(L1),L1)
3     I=SPACE(I+1)
5      .NOT. (SPACE(I+1) .EQ. 0)
COND SPACE(I) .NE. ELT(LENGTH(L1),L1)
                                                  ((0 .NE. ELT(LENGTH(L1),L1)-SPACE(I)) .AND.
                                                  (0 .NE. SPACE(1+SPACE(1+I))))
                                                   .IMPLIES.
                                                  ((0 .NE.
                                                  ELT(LENGTH(L1),L1)-SPACE(SPACE(1+I))))

7
COND SPACE(I)=ELT(LENGTH(L1),L1)
4     SPACE(I+1)=FREE
      FREE=LINE
      LINE=0
COND LV(LINE,L1) .AND. L1=NIL
                                                  ((0 .EQ. ELT(LENGTH(L1),L1)-SPACE(I)))
                                                   .IMPLIES.
                                                  ((LV(0,L1)) .AND. (0 .EQ. -L1+NIL))
```

2.10) DELETE(LINE)

DELETE deletes the entire string LINE and returns it to the string of available space.

example: LINE = "NOTHEEND"
result : LINE = 0 and the string of available space is nine cells longer.

VERIFICATION:

Path 1: insufficient assertion or verification conditions.

Path 2: Since LINE=0 hence L1=nil.

Path 3: Since SPACE(LINE+1)=0 and LV(LINE,L1), hence SPACE(LINE)=ELT(LENGTH(L1),L1).

Path 4: SPACE(LINE+1)≠0, therefore SPACE(LINE)≠ ELT(LENGTH(L1),L1).

Path 5: Similar to path 3.

Path 6: Similar to path 4.

Path 7: insufficient verification conditions.

```
CC     LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC     SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC     .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC     LIST.PIR(P,L)=EXISTS(P=ADDR(K,L),K,1,LENGTH(L)).
CC     LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC     ELT(I,L)=CAR(CDRN(I-1,L)).
CC     CDRN(0,L)=L.
CC     CDRN(I,L)=CDR(CDRN(I-1,L)).
CC     LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1)=LV(XX,L1),
CC    (IF SPACE(I)=YY THEN SPACE(I)=SPACE(ZZ+1),FREE=YY)
CC     .OR. (IF XX=YY, THEN XX=SPACE(ZZ+1), FREE=YY)
CC     .OR. (IF ZZ=0, IF SPACE(I)=YY, THEN SPACE(I)=0, FREE=YY)
COND    LV(XX,L1)
       SUBROUTINE DELETE(XX,YY,ZZ)
       COMMON/BLOCK/SPACE(10000),FAIL,FREE
       INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ
C*************************************************************************
C  SUBROUTINE WILL DELETE STRING XX FROM YY TO ZZ INCLUSIVELY.           *
C  BESIDES IT WILL DO THE FOLLOWING:                                     *
C      (1) IF ZZ=0, THEN IT WILL DELETE FROM YY TO THE END OF THE STRING.*
C      (2) IF YY=ZZ, ONE ELEMENT IS DELETED.                             *
C      (3) IF YY=ZZ=0, NOTHING HAPPENS.                                  *
C      (4) IF YY=0 BUT NOT ZZ, THEN ERROR MASSAGE IS PRINTED.            *
C*************************************************************************M*
       IF (XX.EQ.0) GO TO 21
    9  IF (YY.EQ.0.0) GO TO 20
       POINTE=XX
       IF (XX.EQ.YY) GO TO 15
   18  IF (SPACE(POINTE+1).EQ.YY) GO TO 17
       POINTE = SPACE(POINTE+1)
COND    POINTE .NE. YY,XX .NE. YY, LIST.PIR(POINTE,L1)
       GO TO 18
   17  IF (ZZ.EQ.0) GO TO 14
       KK =  YY
       SPACE(POINTE+1) = SPACE(ZZ+1)
   16  SPACE(ZZ+1) = FREE
       FREE = KK
COND   LV(XX,L1),L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
       RETURN
COND    XX=YY, LIST.PIR(POINTE,L1)
   15  IF (ZZ.EQ.0) GO TO 10
C  IN CASE XX=YY, THE ORIGINAL VALUE OF YY NEEDS TO BE ORESERVED.
C  HENCE LET KK=
       KK =  YY
       XX = SPACE(ZZ+1)
       GO TO 16
COND    ZZ=0,XX=YY, LIST.PIR(POINTE,L1)
   10  XX = 0
       TEMP = FREE
       FREE = POINTE
       GO TO 11
COND    ZZ=0,XX .NE. YY, LIST.PIR(POINTE,L1),SPACE(POINTE+1)=YY
   14  TEMP = FREE
       FREE = SPACE(POINTE+1)
```

```
      SPACE(POINTE+1)  = 0                                                    1
      POINTE = FREE
   11 IF (SPACE(POINTE+1)) 12,13,12
COND   SPACE(POINTS) .NE. ELT(LENGTH(L1),L1),LV(XX,L1)
COND          L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
   12 POINTE =SPACE(POINTE+1)
      GO TO 11                                                                2
COND   SPACE(POINTS)=ELT(LENGTH(L1),L1),LV(XX,L1),
COND          L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
   13 SPACE(POINTE+1) = TEMP
COND  (LV(XX,L1),L1=NIL) .OR. (LV(XX,L1),
COND          L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1)))
   21 RETURN                                                                  3
   20 IF (ZZ.EQ.0) GO TO21
      WRITE(6,22)
   22 FORMAT(1X,45H.....ERROR IN SECOND PARAMETER OF DELETE.....     )
COND   ERROR(DELETE)
      STOP
      END
```

	PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
	COND LV(XX,L1)		
	SUBROUTINE DELETE(XX,YY,ZZ)		
	COMMON/BLOCK/SPACE(10000),FAIL,FREE		
	INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ		
	IF(XX .EQ. 0)GO TO 21		
9	IF(YY .EQ. 0)GO TO 20		
	POINTE=XX	[POINTE]	[XX]
	IF(XX .EQ. YY)GO TO 15		
18	IF(SPACE(POINTE+1) .EQ. YY)GO TO 17		
	POINTE=SPACE(POINTE+1)	[POINTE]	[SPACE,POINTE]
	COND POINTE .NE. YY .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1)		
	GO TO 18		
17	IF(ZZ .EQ. 0)GO TO 14		
	KK=YY	[KK]	[YY]
	SPACE(POINTE+1)=SPACE(ZZ+1)	[SPACE]	[SPACE,POINTE,ZZ]
16	SPACE(ZZ+1)=FREE	[SPACE]	[SPACE,ZZ,FREE]
	FREE=KK	[FREE]	[KK]
	COND LV(XX,L1) .AND. L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))		
	RETURN		
	COND XX=YY .AND. LIST.PIR(POINTE,L1)		
15	IF(ZZ .EQ. 0)GO TO 10		
	KK=YY	[KK]	[YY]
	XX=SPACE(ZZ+1)	[XX]	[SPACE,ZZ]
	GO TO 16		
	COND ZZ=0 .AND. XX=YY .AND. LIST.PIR(POINTE,L1)		
10	XX=0	[XX]	[]
	TEMP=FREE	[TEMP]	[FREE]
	FREE=POINTE	[FREE]	[POINTE]
	GO TO 11		
	COND ZZ=0 .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1) .AND.		
	COND SPACE(POINTE+1)=YY		
14	TEMP=FREE	[TEMP]	[FREE]
	FREE=SPACE(POINTE+1)	[FREE]	[SPACE,POINTE]
	SPACE(POINTE+1)=0	[SPACE]	[SPACE,POINTE]
	POINTE=FREE	[POINTE]	[FREE]
11	IF(SPACE(POINTE+1) .EQ. 0)GO TO 13		
3	COND SPACE(POINTE) .NE. ELT(LENGTH(L1),L1) .AND.		
	COND LV(XX,L1)L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))		
12	POINTE=SPACE(POINTE+1)	[POINTE]	[SPACE,POINTE]
	GO TO 11		
	COND SPACE(POINTE)=ELT(LENGTH(L1),L1) .AND. LV(XX,L1) .AND.		
2	COND L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))		
13	SPACE(POINTE+1)=TEMP	[SPACE]	[SPACE,POINTE,TEMP]
	COND (LV(XX,L1),L1 .EQ. NIL) .OR. (LV(XX,L1),L1 .EQ.		
	COND LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1)))		
21	RETURN		
20	IF(ZZ .EQ. 0)GO TO 21		
1	WRITE(6,22)		
22	FORMAT(1X,45H.....ERROR IN SECOND PARAMETER OF DELETE.....		

```
COND ERROR(DELETE)
      STOP
      END
```

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(XX,L1) SUBROUTINE DELETE(XX,YY,ZZ) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ (XX .EQ. 0) COND (LV(XX,L1),(L1 .EQ. NIL) .OR. (LV(XX,L1),L1 .EQ. COND LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1)))	((LV(XX,L1)) .AND. (0 .EQ. XX)) .IMPLIES. ((0 .EQ. -LV(XX,L1),L1≠NIL) .OR. (0 .EQ. LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE (1+ZZ),L1))-LV(XX,L1),L1))
2	COND LV(XX,L1) SUBROUTINE DELETE(XX,YY,ZZ) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ .NOT. (XX .EQ. 0) 9 (YY .EQ. 0) 20 (ZZ .EQ. 0) COND (LV(XX,L1),(L1 .EQ. NIL) .OR. (LV(XX,L1),(L1 .EQ. COND LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1)))	((LV(XX,L1)) .AND. (0 .NE. XX) .AND. (0 .EQ. YY) .AND. (0 .EQ. ZZ)) .IMPLIES. ((0 .EQ. -LV(XX,L1),L1≠NIL) .OR. (0L1 .EQ. LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPAC (1+ZZ),L1)),LV(XX,L1)=))
3	COND LV(XX,L1) SUBROUTINE DELETE(XX,YY,ZZ) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ .NOT. (XX .EQ. 0) 9 (YY .EQ. 0) 20 .NOT. (ZZ .EQ. 0) WRITE(6,22) 22 FORMAT(1X,45H.....ERROR IN SECOND PARAMETER OF DELETE..... COND ERROR(DELETE)	((LV(XX,L1)) .AND. (0 .NE. XX) .AND. (0 .EQ. YY) .AND. (0 .NE. ZZ)) .IMPLIES. ((ERROR(DELETE)))

```
4
      COND LV(XX,L1)
            SUBROUTINE DELETE(XX,YY,ZZ)
            COMMON/BLOCK/SPACE(10000),FAIL,FREE
            INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ
             .NOT. (XX .EQ. 0)
      9      .NOT. (YY .EQ. 0)
            POINTE=XX
            (XX .EQ. YY)
      COND XX=YY .AND. LIST.PIR(POINTE,L1)
                                                  ((LV(XX,L1)) .AND. (0 .NE. XX) .AND.
                                                  (0 .NE. YY) .AND. (0 .EQ. -XX+YY))
                                                   .IMPLIES.
                                                  ((LIST.PIR(XX,L1)))

5
      COND LV(XX,L1)
            SUBROUTINE DELETE(XX,YY,ZZ)
            COMMON/BLOCK/SPACE(10000),FAIL,FREE
            INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ
             .NOT. (XX .EQ. 0)
      9      .NOT. (YY .EQ. 0)
            POINTE=XX
             .NOT. (XX .EQ. YY)
      18    (SPACE(POINTE+1) .EQ. YY)
      17    (ZZ .EQ. 0)
      COND ZZ=0 .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1) .AND. SPACE(POINTE+1)=YY
                                                  ((LV(XX,L1)) .AND. (0 .NE. XX) .AND.
                                                  (0 .NE. YY) .AND. (0 .NE. -XX+YY)
                                                  .AND. (0 .EQ. -SPACE(1+XX)+YY) .AND.
                                                  (0 .EQ. ZZ))
                                                   .IMPLIES.
                                                  ((LIST.PIR(XX,L1)) .AND. (0 .NE.
                                                  -XX+YY))

6
      COND LV(XX,L1)
            SUBROUTINE DELETE(XX,YY,ZZ)
            COMMON/BLOCK/SPACE(10000),FAIL,FREE
            INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ
             .NOT. (XX .EQ. 0)
      9      .NOT. (YY .EQ. 0)
            POINTE=XX
             .NOT. (XX .EQ. YY)
      18    (SPACE(POINTE+1) .EQ. YY)
      17     .NOT. (ZZ .EQ. 0)
            KK=YY
            SPACE(POINTE+1)=SPACE(ZZ+1)
      16    SPACE(ZZ+1)=FREE
            FREE=KK
      COND LV(XX,L1) .AND. L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                  ((LV(XX,L1)) .AND. (0 .NE. XX) .AND.
                                                  (0 .NE. YY) .AND. (0 .NE. -XX+YY)
```

```
                                                                      .AND. (0 .EQ. -SPACE(1+XX)+YY) .AND.
                                                                      (0 .NE. ZZ))
                                                                       .IMPLIES.
                                                                      ((LV(XX,L1)) .AND. (0 .EQ.
                                                                      -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,F
                                                                      REE,0))))
                                                                          L1
      7
            COND (V(XX,L1)
                  SUBROUTINE DELETE(XX,YY,ZZ)
                  COMMON/BLOCK/SPACE(10000),FAIL,FREE
                  INTEGER XX,YY,ZZ,TEMP,SPACE,FAIL,FREE,POINTE,ZZZ
                   .NOT. (XX .EQ. 0)
            9      .NOT. (YY .EQ. 0)
                  POINTE=XX
                   .NOT. (XX .EQ. YY)
            18     .NOT. (SPACE(POINTE+1) .EQ. YY)
                  POINTE=SPACE(POINTE+1)
            COND POINTE .NE. YY .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1)
                                                                      ((LV(XX,L1)) .AND. (0 .NE. XX) .AND.
                                                                      (0 .NE. YY) .AND. (0 .NE. -XX+YY)
                                                                      .AND. (0 .NE. -SPACE(1+XX)+YY))
                                                                       .IMPLIES.
                                                                      ((LIST.PIR(SPACE(1+XX),L1)) .AND. (0
                                                                      .NE. -XX+YY) .AND. (0 .NE.
                                                                      -SPACE(1+XX)+YY))

      8
            COND POINTE .NE. YY .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1)
            18    (SPACE(POINTE+1) .EQ. YY)
            17    (ZZ .EQ. 0)
            COND ZZ=0 .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1) .AND. SPACE(POINTE+1)=YY
                                                                      ((0 .NE. -POINTE+YY) .AND. (0 .NE.
                                                                      -XX+YY) .AND. (LIST.PIR(POINTE,L1))
                                                                      .AND. (0 .EQ. -SPACE(1+POINTE)+YY)
                                                                      .AND. (0 .EQ. ZZ))
                                                                       .IMPLIES.
                                                                      ((LIST.PIR(POINTE,L1)) .AND. (0 .NE.
                                                                      -XX+YY))

      9
            COND POINTE .NE. YY .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1)
            18    (SPACE(POINTE+1) .EQ. YY)
            17     .NOT. (ZZ .EQ. 0)
                  KK=YY
                  SPACE(POINTE+1)=SPACE(ZZ+1)
            16    SPACE(ZZ+1)=FREE
                  FREE=KK
            COND LV(XX,L1) .AND. L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                                      ((0 .NE. -POINTE+YY) .AND. (0 .NE.
                                                                      -XX+YY) .AND. (LIST.PIR(POINTE,L1))
                                                                      .AND. (0 .EQ. -SPACE(1+POINTE)+YY)
                                                                      .AND. (0 .NE. ZZ))
```

```
                                                                .IMPLIES.
                                                                ((LV(XX,L1)) .AND. (0 .EQ.
                                                                -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,F
                                                                REE,L1))))

10
        COND POINTE .NE. YY .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1)
        18     .NOT. (SPACE(POINTE+1) .EQ. YY)
               POINTE=SPACE(POINTE+1)
        COND POINTE .NE. YY .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1)
                                                                ((0 .NE. -POINTE+YY) .AND. (0 .NE.
                                                                -XX+YY) .AND. (LIST.PIR(POINTE,L1))
                                                                .AND. (0 .NE. -SPACE(1+POINTE)+YY))
                                                                 .IMPLIES.
                                                                ((LIST.PIR(SPACE(1+POINTE),L1))
                                                                .AND. (0 .NE. -XX+YY) .AND. (0 .NE.
                                                                -SPACE(1+POINTE)+YY))

11
        COND XX=YY .AND. LIST.PIR(POINTE,L1)
        15     (ZZ .EQ. 0)
        COND ZZ=0 .AND. XX=YY .AND. LIST.PIR(POINTE,L1)
                                                                ((0 .EQ. -XX+YY) .AND.
                                                                (LIST.PIR(POINTE,L1)) .AND. (0 .EQ.
                                                                ZZ))
                                                                 .IMPLIES.
                                                                ((LIST.PIR(POINTE,L1)))

12
        COND XX=YY .AND. LIST.PIR(POINTE,L1)
        15     .NOT. (ZZ .EQ. 0)
               KK=YY
               XX=SPACE(ZZ+1)
        16     SPACE(ZZ+1)=FREE
               FREE=KK
        COND LV(XX,L1) .AND. L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                                ((0 .EQ. -XX+YY) .AND.
                                                                (LIST.PIR(POINTE,L1)) .AND. (0 .NE.
                                                                ZZ))
                                                                 .IMPLIES.
                                                                ((LV(SPACE(1+ZZ),L1)) .AND. (0 .EQ.
                                                                -L1+LISTDIFFER(LV(SPACE(1+ZZ),L1),SUB
                                                                LIST(YY,FREE,L1))))

13
        COND ZZ=0 .AND. XX=YY .AND. LIST.PIR(POINTE,L1)
        10     XX=0
               TEMP=FREE
               FREE=POINTE
        11     (SPACE(POINTE+1) .EQ. 0)
        COND SPACE(POINTE)=ELT(LENGTH(L1),L1) .AND. LV(XX,L1) .AND.
        COND L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
```

```
                                                                    ((0 .EQ. ZZ) .AND. (0 .EQ. -XX+YY)
                                                                    .AND. (LIST.PIR(POINTE,L1)) .AND. (0
                                                                    .EQ. SPACE(1+POINTE)))
                                                                     .IMPLIES.
                                                                    ((LV(0,L1)) .AND. (0 .EQ.
                                                                    -L1+LISTDIFFER(LV(0,L1),SUBLIST(YY,SP
                                                                    ACE(1+ZZ),L1))) .AND. (0 .EQ.
                                                                    ELT(LENGTH(L1),L1)-SPACE(POINTE)))

    14
        COND ZZ=0 .AND. XX=YY .AND. LIST.PIR(POINTE,L1)
        10    XX=0
              TEMP=FREE
              FREE=POINTE
        11     .NOT. (SPACE(POINTE+1) .EQ. 0)
        COND SPACE(POINTE) .NE. ELT(LENGTH(L1),L1) .AND.
        COND LV(XX,L1)L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                                    ((0 .NE.
                                                                    ELT(LENGTH(L1),L1)-SPACE(POINTE)))
                                                                     .IMPLIES.
                                                                    ((LV(0,L1)) .AND. (0 .EQ.
                                                                    -L1+LISTDIFFER(LV(0,L1),SUBLIST(YY,SP
                                                                    ACE(1+ZZ),L1))))

    15
        COND ZZ=0 .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1) .AND. SPACE(POINTE+1)=YY
        14    TEMP=FREE
              FREE=SPACE(POINTE+1)
              SPACE(POINTE+1)=0
              POINTE=FREE
        11    (SPACE(POINTE+1) .EQ. 0)
        COND SPACE(POINTE)=ELT(LENGTH(L1),L1) .AND. LV(XX,L1) .AND.
        COND L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                                    ((0 .EQ. ZZ) .AND. (0 .NE. -XX+YY)
                                                                    .AND. (LIST.PIR(POINTE,L1)) .AND. (0
                                                                    .EQ. -SPACE(1+POINTE)+YY) .AND. (0
                                                                    .EQ. SPACE(1+SPACE(1+POINTE))))
                                                                     .IMPLIES.
                                                                    ((LV(XX,L1)) .AND. (0 .EQ.
                                                                    -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,S
                                                                    PACE(1+ZZ),L1))) .AND. (0 .EQ.
                                                                    ELT(LENGTH(L1),L1)-SPACE(SPACE(1+POIN
                                                                    TE))))

    16
        COND ZZ=0 .AND. XX .NE. YY .AND. LIST.PIR(POINTE,L1) .AND. SPACE(POINTE+1)=YY
        14    TEMP=FREE
              FREE=SPACE(POINTE+1)
              SPACE(POINTE+1)=0
              POINTE=FREE
        11     .NOT. (SPACE(POINTE+1) .EQ. 0)
        COND SPACE(POINTE) .NE. ELT(LENGTH(L1),L1) .AND.
        COND LV(XX,L1)L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
```

```
                                                                ((0 .NE.
                                                                ELT(LENGTH(L1),L1)-SPACE(SPAC (1+POIN
                                                                TE))))
                                                                 .IMPLIES.
                                                                ((LV(XX,L1)) .AND. (0 .EQ.
                                                                -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,S
                                                                PACE(1+ZZ),L1))))

17
      COND SPACE(POINTE) .NE. ELT(LENGTH(L1),L1) .AND.
      COND LV(XX,L1)L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
      12    POINTE=SPACE(POINTE+1)
      11    (SPACE(POINTE+1) .EQ. 0)
      COND SPACE(POINTE)=ELT(LENGTH(L1),L1) .AND. LV(XX,L1) .AND.
      COND L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                                ((LV(XX,L1)) .AND. (0 .EQ.
                                                                -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,S
                                                                PACE(1+ZZ),L1))) .AND. (0 .EQ.
                                                                SPACE(1+SPACE(1+POINTE))))
                                                                 .IMPLIES.
                                                                ((LV(XX,L1)) .AND. (0 .EQ.
                                                                ELT(LENGTH(L1),L1)-SPACE(SPACE(1+POIN
                                                                TE))))

18
      COND SPACE(POINTE) .NE. ELT(LENGTH(L1),L1) .AND.
      COND LV(XX,L1)L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
      12    POINTE=SPACE(POINTE+1)
      11     .NOT. (SPACE(POINTE+1) .EQ. 0)
      COND SPACE(POINTE) .NE. ELT(LENGTH(L1),L1) .AND.
      COND LV(XX,L1)L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
                                                                ((0 .NE.
                                                                ELT(LENGTH(L1),L1)-SPACE(SPAC (1+POIN
                                                                TE))))
                                                                 .IMPLIES.
                                                                ((LV(XX,L1)) .AND. (0 .EQ.
                                                                -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,S
                                                                PACE(1+ZZ),L1))))

19
      COND SPACE(POINTE)=ELT(LENGTH(L1),L1) .AND. LV(XX,L1) .AND.
      COND L1=LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1))
      13    SPACE(POINTE+1)=TEMP
      COND (LV(XX,L1),L1 .EQ. NIL) .OR. (LV(XX,L1),L1 .EQ.
      COND LISTDIFFER(LV(XX,L1),SUBLIST(YY,SPACE(ZZ+1),L1)))
                                                                ((0 .EQ.
                                                                ELT(LENGTH(L1),L1)-SPACE(POINTE))
                                                                .AND. (LV(XX,L1)) .AND. (0 .EQ.
                                                                -L1+LISTDIFFER(LV(XX,L1),SUBLIST(YY,S
                                                                PACE(1+ZZ),L1))))
                                                                 .IMPLIES.
                                                                ((0 .EQ. -LV(XX,L1),L1+NIL) .OR. (0
                                                                .EQ.
```

```
LISTDIFF=(LV(XX,L1),SUBLIST(YY,SPACE
(1+ZZ),L1))-LV(XX,L1),L1))
```

2.11) DELETE(XX,YY,ZZ)

The characters at and between the pointers YY and ZZ are deleted from the string XX.

VERIFICATION:

Path 1: Since XX=0, L1=nil.

Path 2: insufficient verification condition.

Path 3: Since WRITE statement has not been implemented there is not enough verification condition.

Path 4: Since LV(XX,L1) therefore LIST.PIR(XX,L1).

Path 5: XX≠YY is not changed in the path.

Path 6: insufficient verification condition.

Path 7: SPACE(XX+1)≠YY and XX≠YY are not changed. Since LV(XX,L1), hence LIST.PIR(SPACE(XX+1),L1).

Path 8: XX≠YY is not changed.

Path 9: insufficient assertion.

Path 10: XX≠YY and SPACE(POINTE+1)≠YY are not changed.

Path 11: LIST.PIR(POINTE,L1) is not changed.

Path 12: insufficient assertion.

Path 13: insufficient assertion.

Path 14: insufficient assertion

Path 15: insufficient assertion;

Path 16: insufficient assertion.

Path 17: Since SPACE(SPACE(POINTE+1)+1)=0, hence SPACE(SPACE(POINTE+1))=ELT(LENGTH(L1),L1).

Path 18: insufficient assertion

Path 19: insufficient assertion.

```
CC      LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC      SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC      SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC      .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC      CDRN(0,L)=L.
CC      CDRN(I,L)=CDR(CDRN(I-1,L)).
CC      ELT(I,L)=CAR(CDRN(I-1,L)).
COND     LV(I,L1)
        SUBROUTINE DUP(I,J,K)
        COMMON/BLOCK/IFREE(10000),FAIL,FREE
           INTEGER FAIL,FREE
C THIS COPIES FROM INDEX J  UP TO BUT NOT INCLUDING IND X K INTO THE STRING I
        IF(K) 4,5,4
      4 KK=K
          JJ = J
          I=0
          N = 0
1         IF(JJ-KK)3,2,3
COND     LV(I,L1),SUBLIST(J,JJ,L1),L1=SL1,KK=K
3         CALL INSERT(I,N,IFREE(JJ))
COND    ELT(N+1,L1)=IFREE(JJ),CAR(L1)=IFREE(J),KK=K,
COND     LV(I,L1),SUBLIST(J,IFREE(JJ+1),L1),L1=SL1
          JJ = IFREE(JJ+1)
          N = N+ 1
          GO TO 1
      5 I=0
COND     (LV(I,L1),(L1=NIL)) .OK.
COND     (LV(I,L1),SUBLIST(J,K,L1),(L1=SL1))
2         RETURN
        END
```

	PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
	COND LV(I,LI)		
	SUBROUTINE DEP(I,J,K)		
	COMMON/BLOCK/IFREE(1000),FAIL,FREE		
	INTEGER FAIL,FREE		
	IF(K .EQ. 0)GO TO 5		
4	KK=K	[KK]	[K]
	JJ=J	[JJ]	[J]
	I=0	[I]	[]
	N=0	[N]	[]
1	IF(JJ=KK .EQ. 0)GO TO 2		
	COND LV(I,LI) .AND. SUBLIST(J,JJ,LI) .AND. LI=SL1 .AND. KK=K		
3	CALL PASS1(I,N,IFREE(JJ))		
	COND SLT(N+1,LI)=IFREE(JJ) .AND. CAP(LI)=IFREE(J) .AND. KK=K .AND.		
	COND LV(I,LI) .AND. SUBLIST(J,IFREE(JJ+1),LI) .AND. LI=SL1		
	JJ=IFREE(JJ+1)	[JJ]	[IFREE,JJ]
	N=N+1	[N]	[N]
	GO TO 1		
5	I=0	[I]	[]
	COND (LV(I,LI),(LI .EQ. NIL)) .OR. (LV(I,LI),SUBLIST(J,K,LI),(LI .EQ.		
	COND SL1))		
2	RETURN		
	END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(I,L1) SUBROUTINE JUP(I,J,K) COMMON/BLOCK/IFREE(10000),FAIL,FREE INTEGER FAIL,FREE (K .E). 0) 5 I=0 COND (LV(I,L1),(L1 .EQ. NIL)) .OR. (LV(I,L1),SUBLIST(J,K,L1),(L1 .EQ. SL1))	((LV(I,L1)) .AND. (J .EQ. K)) .IMPLIES. ((LV(0,L1),0 .EQ. -L1+NIL) .OR. (LV(0,L1),SUBLIST(J,K,L1),0 .EQ. -L1+SL1))
2	COND LV(I,L1) SUBROUTINE JUP(I,J,K) COMMON/BLOCK/IFREE(10000),FAIL,FREE INTEGER FAIL,FREE .OR. (K .EQ. 0) 4 KK=K JJ=J I=J N=0 1 (JJ-KK .EQ. 0) COND (LV(I,L1),(L1 .EQ. NIL)) .OR. (LV(I,L1),SUBLIST(J,K,L1),(L1 .EQ. SL1))	((LV(I,L1)) .AND. (J .NE. K) .AND. (J .EQ. J-K)) .IMPLIES. ((LV(0,L1),0 .EQ. -L1+NIL) .OR. (LV(0,L1),SUBLIST(J,K,L1),0 .EQ. -L1+SL1))
3	COND LV(I,L1) SUBROUTINE JUP(I,J,K) COMMON/BLOCK/IFREE(10000),FAIL,FREE INTEGER FAIL,FREE .OR. (K .EQ. 0) 4 KK=K JJ=J I=0 N=0 1 .OR. (JJ-KK .EQ. 0) COND LV(I,L1) .AND. SUBLIST(J,JJ,L1) .AND. L1=SL1 .AND. KK=K	((LV(I,L1)) .AND. (J .NE. K) .AND. (J .NE. J-K)) .IMPLIES. ((LV(0,L1)) .AND. (SUBLIST(J,J,L1))

```
                                                                .AND. (0 .EQ. -L1+SL1))                                    1

4
      COND LV(I,L1) .AND. SUBLIST(J,JJ,L1) .AND. L1=SL1 .AND. KK=K
     3       CALL INSERT(I,N,IFREE(JJ))
      COND ELT(1+1,L1)=IFREE(JJ) .AND. CAR(L1)=IFREE(J) .AND. KK=K .AND. LV(I,L1)                                          2
      COND .AND. SUBLIST(J,IFREE(JJ+1),L1) .AND. L1=SL1
                                                                ((LV(I,L1)) .AND. (SUBLIST(J,JJ,L1))
                                                                .AND. (0 .EQ. -L1+SL1) .AND. (0 .EQ.
                                                                K-KK))
                                                                 .IMPLIES.
                                                                ((LV(I,L1)) .AND.
                                                                (SUBLIST(J,IFREE(1+JJ),L1) .AND. (0                        3
                                                                .EQ. -CAR(L1)+IFREE(J)) .AND. (0
                                                                .EQ. -ELT(1+N,L1)+IFREE(JJ)))

5
      COND ELT(N+1,L1)=IFREE(JJ) .AND. CAR(L1)=IFREE(J) .AND. KK=K .AND. LV(I,L1)
      COND .AND. SUBLIST(J,IFREE(JJ+1),L1) .AND. L1=SL1
           JJ=IFREE(JJ+1)
           N=N+1
     1     (JJ-KK .EQ. 0)
      COND (LV(I,L1),(L1 .EQ. NIL)) .OR. (LV(I,L1),SUBLIST(J,K,L1),(L1 .EQ. SL1))
                                                                ((0 .EQ. -ELT(1+N,L1)+IFREE(JJ))
                                                                .AND. (0 .EQ. -CAR(L1)+IFREE(J))
                                                                .AND. (0 .EQ. K-KK) .AND. (LV(I,L1))
                                                                .AND. (SUBLIST(J,IFREE(1+JJ),L1))
                                                                .AND. (0 .EQ. -L1+SL1) .AND. (0 .EQ.
                                                                IFREE(1+JJ)-KK))
                                                                 .IMPLIES.
                                                                ((LV(I,L1),0 .EQ. -L1+NIL) .OR.
                                                                (LV(I,L1),SUBLIST(J,K,L1),0 .EQ.
                                                                -L1+SL1))

6
      COND ELT(N+1,L1)=IFREE(JJ) .AND. CAR(L1)=IFREE(J) .AND. KK=K .AND. LV(I,L1)
      COND .AND. SUBLIST(J,IFREE(JJ+1),L1) .AND. L1=SL1
           JJ=IFREE(JJ+1)
           N=N+1
     1       .NOT. (JJ-KK .EQ. 0)
      COND LV(I,L1) .AND. SUBLIST(J,JJ,L1) .AND. L1=SL1 .AND. KK=K
                                                                ((0 .EQ. -ELT(1+N,L1)+IFREE(JJ))
                                                                .AND. (0 .EQ. -CAR(L1)+IFREE(J))
                                                                .AND. (0 .EQ. K-KK) .AND. (LV(I,L1))
                                                                .AND. (SUBLIST(J,IFREE(1+JJ),L1))
                                                                .AND. (0 .EQ. -L1+SL1) .AND. (0 .NE.
                                                                IFREE(1+JJ)-KK))
                                                                 .IMPLIES.
                                                                ((LV(I,L1)) .AND.
                                                                (SUBLIST(J,IFREE(1+JJ),L1)))
```

2.12) DUP(I,J,K)

This subroutine is the same as COPPY but returns I=0 if K=0.

VERIFICATION:

Path 1: insufficient assertion.

Path 2: Since LV(0,L1) and SUBLIST(J,K,L1) and J=K, hence L1=SL1.

Path 3: Similar to path 2.

Path 4: Since the call statement has not been taken into account, assertions here are not sufficient to be verified,

Path 5: obvious.

```
CC  EXP(N,I)=N**I
CC  LENGTH(L) = (IF NIL THEN 0 ELSE LENGTH(CDR(L))+1)
C   LV(LP,L) MEANS THE LIST VALUE OF LP IS L
CC  LV(LP,L) = (EF(K,(1..N)), ((1.LE.K(I), K(I).LE.5000,
CC  SPACE(2*K(I)-1) = L(I), SPACE(2*K(I)) = 2*K(I+1) -1),
CC  I .IN. (1..N)), LP = 2*K(1) - 1)
C   CNTLEN(NUMBER) - COUNTING THE LENGTH (NUMBER OF SIGNIFICANT
C   DIGITS) OF NUMBER
CC  CNTLEN(NUMBER)= (IF NUMBER.LE.9 THEN 1 ELSE CNTLEN(NUMBER/10)+1)
C   MSD(NUMBER) - MOST SIGNIFICANT DIGIT OF NUMBER
CC  MSD(NUMBER)= (IF NUMBER.LE.9 THEN NUMBER ELSE MSD(NUMBER/10))
CC  IALPHN(27)=1H0, IALPHN(28)=1H1, IALPHN(29)=1H2, IALPHN(30)=1H3,
CC  IALPHN(31)=1H4, IALPHN(32)=1H5, IALPHN(33)=1H6, IALPHN(34)=1H7,
CC  IALPHN(35)=1H8, IALPHN(36)=1H9
CC  TYPE(L,1)=*REAL*, TYPE(L,2)=*INTEGER*, TYPE(L,3)=*STRING*
CC  CTYPE(NO,*STRING*) = IALPHN(NO+27), 0.LE.NO,NO.LE.9
C
COND  TYPE(NUMBER,2), NUMBER=NO
      INTEGER FUNCTION ENCODE(NUMBER)
      COMMON/ALPHNU/IALPHN(63)
      COMMON/BLOCK/SPACE(10000),IFQ,IFQQ
      INTEGER SPACE,TNUM,NUMBER,COUNT,TEMP,ADDRES,NTEN
C  THIS FUNCTION CONVERTS A NUMBER INTO A STRING , LEAVING OFF LEADING
C  ZEROS
      IFLAG=0
      IF(NUMBER.GE.0)GO TO 8
      IFLAG=1
      NUMBER=-NUMBER
8     ENCODE=NEWCEL(0)
      ADDRES=ENCODE
COND  NUMBER .GE. 0, (IFLAG=0, NO.GE.0), (IFLAG=1, NO.LT.0),
COND  ADDRES= ENCODE= AO
      TNUM=NUMBER
    3 INUM=TNUM
C  COUNT THE NUMBER OF SIGNIFICANT DIGITS IN NUMBER
      COUNT=1
COND  (COUNT=1 .IN. INUM=TNUM) .OR. (COUNT= CNTLEN(INUM)-CNTLEN(TNUM)+1)
    1 IF(TNUM-9) 4,4,2
    2 COUNT=COUNT+1
      TNUM=TNUM/10
      GO TO 1
4     SPACE(ADDRES)=IALPHN(TNUM+27)
COND  COUNT= CNTLEN(INUM), TNUM= MSD(INUM),
COND  SPACE(ADDRES)= CTYPE(TNUM, *STRING*), ADDRES= AO
      COUNT=COUNT-1
      IF(COUNT) 6,7,6
    6 TEMP=ADDRES
      ADDRES=NEWCEL(0)
      SPACE(TEMP+1)=ADDRES
COND  (SPACE(AO+1)= ADDRES, COUNT.NE.0)
      NTEN=1
      I = 1
COND  NTEN = EXP(10,(I-1))
    5 NTEN=NTEN*10
      I = I + 1
```

```
      IF ( I .LE. COUNT ) GO TO 5
      TNUM=INUM-TNUM*NTEN
COND  TNUM= INUM-MSD(INUM)*NTEN, NTEN= EXP(10,(COUNT-L))
      GO TO 3
    7 SPACE(ADDRES+1)=0
COND  LV(ENCODE,L1), ELT(LENGTH(L1), L1)= SPACE(ADDRES),
COND ((IFLAG=0, NO.GE.0) .OR. (IFLAG=1, NO.LT.0))
      IF (IFLAG .EQ. 0) GO TO 9
      NUMBER=-NUMBER
      CALL INSERT(ENCODE,0,IALPHN(38))
COND  LV(ENCODE,L), ((L=L1,IFLAG=0) .OR. (ELT(1,L)=1H-,
COND  COR(L)=L1, IFLAG=1)), TYPE(L,3)
9     RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND TYPE(NUMBER,2) .AND. NUMBER=NO		
INTEGER FUNCTION ENCODE(NUMBER)		
COMMON/ALPHNU/IALPHN(63)		
COMMON/BLOCK/SPACE(10000),IFQ,IFQQ		
INTEGER SPACE,TNUM,NUMBER,COUNT,TEMP,ADDRES,NTEN		
IFLAG=0	[IFLAG]	[]
IF(NUMBER .GE. 0)GO TO 8		
IFLAG=1	[IFLAG]	[]
NUMBER=-NUMBER	[NUMBER]	[NUMBER]
8 ENCODE=NEWCEL(0)	[ENCODE]	[NEWCEL]
ADDRES=ENCODE	[ADDRES]	[ENCODE]
COND NUMBER .GE. 0 .AND. ((IFLAG .EQ. 0,NO .GE. 0) .AND. ((IFLAG .EQ.		
COND 1,NO .LT. 0) .AND. ADDRES=ENCODE=AO		
TNUM=NUMBER	[TNUM]	[NUMBER]
3 INUM=TNUM	[INUM]	[TNUM]
COUNT=1	[COUNT]	[]
COND ((COUNT .EQ. 1.IM.INUM .EQ. TNUM) .OR. (COUNT .EQ.		
COND CNTLEN(INUM)-CNTLEN(TNUM)+1)		
1 IF(TNUM-9 .LE. 0)GO TO 4		
2 COUNT=COUNT+1	[COUNT]	[COUNT]
TNUM=TNUM/10	[TNUM]	[TNUM]
GO TO 1		
4 SPACE(ADDRES)=IALPHN(TNUM+27)	[SPACE]	[SPACE,ADDRES,IALPHN,TNUM]
COND COUNT=CNTLEN(INUM) .AND. TNUM=MSD(INUM) .AND.		
COND SPACE(ADDRES)=CTYPE(TNUM,*STRING*) .AND. ADDRES=AO		
COUNT=COUNT-1	[COUNT]	[COUNT]
IF(COUNT .EQ. 0)GO TO 7		
6 TEMP=ADDRES	[TEMP]	[ADDRES]
ADDRES=NEWCEL(0)	[ADDRES]	[NEWCEL]
SPACE(TEMP+1)=ADDRES	[SPACE]	[SPACE,TEMP,ADDRES]
COND (SPACE(AO+1) .EQ. ADDRES,COUNT .NE. 0)		
NTEN=1	[NTEN]	[]
I=1	[I]	[]
COND NTEN=EXP(10,(I-1))		
5 NTEN=NTEN*10	[NTEN]	[NTEN]
I=I+1	[I]	[I]
IF(I .LE. COUNT)GO TO 5		
TNUM=INUM-TNUM*NTEN	[TNUM]	[INUM,TNUM,NTEN]
COND TNUM=INUM-MSD(INUM)*NTEN .AND. NTEN=EXP(10,(COUNT-1))		
GO TO 3		
7 SPACE(ADDRES+1)=0	[SPACE]	[SPACE,ADDRES]
COND LV(ENCODE,L1) .AND. ELT(LENGTH(L1),L1)=SPACE(ADDRES) .AND. ((IFLAG		
COND .EQ. 0,NO .GE. 0) .OR. (IFLAG .EQ. 1,NO .LT. 0))		
IF(IFLAG .EQ. 0)GO TO 9		
NUMBER=-NUMBER	[NUMBER]	[NUMBER]
CALL INSERT(ENCODE,0,IALPHN(38))		
COND LV(ENCODE,L) .AND. ((L .EQ. L1,IFLAG .EQ. 0) .OR. (ELT(1,L) .EQ.		
COND 14-,CDR(L) .EQ. L1,IFLAG .EQ. 1)) .AND. TYPE(L,3)		
9 RETURN		

END

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND TYPE(NUMBER,2) .AND. NUMBER=NO INTEGER FUNCTION ENCODE(NUMBER) COMMON/ALPHNU/IALPHN(63) COMMON/BLOCK/SPACE(10000),IFQ,IFQQ INTEGER SPACE,TNUM,NUMBER,COUNT,TEMP,ADDRES,NTEN IFLAG=0 (NUMBER .GE. 0) 8 ENCODE=NEWCEL(0) ADDRES=ENCODE COND NUMBER .GE. 0 .AND. (IFLAG .EQ. 0,NO .GE. 0) .AND. (IFLAG .EQ. 1,NO .LT. COND 0) .AND. ADDRES=ENCODE=AO	((TYPE(NUMBER,2)) .AND. (0 .EQ. NO-NUMBER) .AND. (0 .LE. NUMBER)) .IMPLIES. ((0 .EQ. AO-NEWCEL(0)) .AND. (0 .LT. -0 .EQ. 1,NO) .AND. (0 .LE. 0 .EQ. 0,NO))
2	COND TYPE(NUMBER,2) .AND. NUMBER=NO INTEGER FUNCTION ENCODE(NUMBER) COMMON/ALPHNU/IALPHN(63) COMMON/BLOCK/SPACE(10000),IFQ,IFQQ INTEGER SPACE,TNUM,NUMBER,COUNT,TEMP,ADDRES,NTEN IFLAG=0 .NOT. (NUMBER .GE. 0) IFLAG=1 NUMBER=-NUMBER 8 ENCODE=NEWCEL(0) ADDRES=ENCODE COND NUMBER .GE. 0 .AND. (IFLAG .EQ. 0,NO .GE. 0) .AND. (IFLAG .EQ. 1,NO .LT. COND 0) .AND. ADDRES=ENCODE=AO	((TYPE(NUMBER,2)) .AND. (0 .EQ. NO-NUMBER) .AND. (0 .LT. -NUMBER)) .IMPLIES. ((0 .EQ. AO-NEWCEL(0)) .AND. (0 .LT. 1-NO) .AND. (0 .LE. 0 .EQ. -1+0,NO))
3	COND NUMBER .GE. 0 .AND. (IFLAG .EQ. 0,NO .GE. 0) .AND. (IFLAG .EQ. 1,NO .LT. COND 0) .AND. ADDRES=ENCODE=AO TNUM=NUMBER 3 INUM=TNUM COUNT=1 COND (COUNT .EQ. 1.IM.INUM .EQ. TNUM) .OR. (COUNT .EQ. COND CNTLEN(INUM)-CNTLEN(TNUM)+1)	NONE

```
4
      COND (COUNT .EQ. 1.IM.INUM .EQ. TNUM) .OR. (COUNT .EQ.
      COND CNTLEN(INUM)-CNTLEN(TNUM)+1)
      1      (TNUM-9 .LE. 0)
      4      SPACE(ADDRES)=IALPHN(TNUM+27)
      COND COUNT=CNTLEN(INUM) .AND. TNUM=MSD(INUM) .AND.
      COND SPACE(ADDRES)=CTYPE(TNUM,*STRING*) .AND. ADDRES=A0
```

```
(((0 .EQ. COUNT+TNUM) .OR. (0 .EQ.
1+CNTLEN(INUM)-CNTLEN(TNUM)-CCUNT))
.AND. (0 .LE. 9-TNUM))
 .IMPLIES.
((0 .EQ. A0-ADDRES) .AND. (0 .EQ.
CTYPE(TNUM,*STRING*)-IALPHN(27+TNUM))
 .AND. (0 .EQ. MSD(INUM)-TNUM) .AND.
(0 .EQ. CNTLEN(INUM)-COUNT))
```

```
5
      COND (COUNT .EQ. 1.IM.INUM .EQ. TNUM) .OR. (COUNT .EQ.
      COND CNTLEN(INUM)-CNTLEN(TNUM)+1)
      1       .NOT. (TNUM-9 .LE. 0)
      2      COUNT=COUNT+1
             TNUM=TNUM/10
      COND (COUNT .EQ. 1.IM.INUM .EQ. TNUM) .OR. (COUNT .EQ.
      COND CNTLEN(INUM)-CNTLEN(TNUM)+1)
```

```
(((0 .EQ. COUNT+TNUM) .OR. (0 .EQ.
1+CNTLEN(INUM)-CNTLEN(TNUM)-CCUNT))
.AND. (0 .LT. -9+TNUM))
 .IMPLIES.
((0 .EQ. 1-IM.INUM+COUNT+((TNUM))
.IMPLIES. ((10))) .OR. (0 .EQ.
CNTLEN(INUM)-CNTLEN(((TNUM))
.IMPLIES. ((10)))-COUNT))
```

```
6
      COND COUNT=CNTLEN(INUM) .AND. TNUM=MSD(INUM) .AND.
      COND SPACE(ADDRES)=CTYPE(TNUM,*STRING*) .AND. ADDRES=A0
            COUNT=COUNT-1
            (COUNT .EQ. 0)
      7     SPACE(ADDRES+1)=0
      COND LV(ENCODE,L1) .AND. ELT(LENGTH(L1),L1)=SPACE(ADDRES) .AND. ((IFLAG .EQ.
      COND 0,NO .GE. 0) .OR. (IFLAG .EQ. 1,NO .LT. 0))
```

```
((0 .EQ. CNTLEN(INUM)-COUNT) .AND.
(0 .EQ. MSD(INUM)-TNUM) .AND. (0
.EQ.
CTYPE(TNUM,*STRING*)-SPACE(ADDRES))
.AND. (0 .EQ. A0-ADDRES) .AND. (0
.EQ. -1+COUNT))
 .IMPLIES.
((LV(ENCODE,L1)) .AND. ((0 .LE. 0
.EQ. -IFLAG+0,NO) .OR. (0 .LT.
IFLAG-NO)) .AND. (0 .EQ.
-ELT(LENGTH(L1),L1)+SPACE(ADDRES)))
```

```
7
COND COUNT=CNTLEN(INUM) .AND. TNUM=MSD(INUM) .AND.
COND SPACE(ADDRES)=CTYPE(TNUM,*STRING*) .AND. ADDRES=AO
       COUNT=COUNT-1
        .NOT. (COUNT .EQ. 0)
6      TEMP=ADDRES
       ADDRES=NEWCEL(0)
       SPACE(TEMP+1)=ADDRES
COND (SPACE(AO+1) .EQ. ADDRES,COUNT .NE. 0)
```

```
((0 .EQ. CNTLEN(INUM)-COUNT) .AND.
(0 .EQ. MSD(INUM)-TNUM) .AND. (0
.EQ.
CTYPE(TNUM,*STRING*)-SPACE(ADDRES))
.AND. (0 .EQ. AO-ADDRES) .AND. (0
.NE. -1+COUNT))
 .IMPLIES.
((0 .NE. 1-COUNT+SPACE(1+AO)))
```

```
8
COND (SPACE(AO+1) .EQ. ADDRES,COUNT .NE. 0)
       NTEN=1
       I=1
COND NTEN=EXP(10,(I-1))
```

```
((0 .NE. SPACE(1+AO)))
 .IMPLIES.
((0 .EQ. -1+EXP(10,0)))
```

```
9
COND NTEN=EXP(10,(I-1))
5      NTEN=NTEN*10
       I=I+1
       (I .LE. COUNT)
COND NTEN=EXP(10,(I-1))
```

```
((0 .EQ. EXP(10,-1+I)-NTEN) .AND. (0
.LE. -1+COUNT-I))
 .IMPLIES.
((0 .EQ. EXP(10,I)-10*NTEN))
```

```
10
COND NTEN=EXP(10,(I-1))
5      NTEN=NTEN*10
       I=I+1
        .NOT. (I .LE. COUNT)
       TNUM=INUM-TNUM*NTEN
COND TNUM=INUM-MSD(INUM)*NTEN .AND. NTEN=EXP(10,(COUNT-1))
```

```
((0 .EQ. EXP(10,-1+I)-NTEN) .AND. (0
.LT. 1-COUNT+I))
 .IMPLIES.
((0 .EQ. EXP(10,-1+COUNT)-10*NTEN)
.AND. (0 .EQ.
-10*MSD(INUM)*NTEN+10*NTEN))
```

```
11
      COND TNUM=INUM-MSD(INUM)*NTEN .AND. NTEN=EXP(10,(COUNT-1))
      3     INUM=TNUM
            COUNT=1
      COND (COUNT .EQ. 1.IM.INUM .EQ. TNUM) .OR. (COUNT .EQ.
      COND CNTLEN(INUM)-CNTLEN(TNUM)+1)
                                                                  NONE

12
      COND LV(ENCODE,L1) .AND. ELT(LENGTH(L1),L1)=SPACE(ADDRES) .AND. ((IFLAG .EQ.
      COND 0,NO .GE. 0) .OR. (IFLAG .EQ. 1,NO .LT. 0))
            (IFLAG .EQ. 0)
      COND LV(ENCODE,L) .AND. ((L .EQ. L1,IFLAG .EQ. 0) .OR. (ELT(1,L) .EQ.
      COND 1H-,CDR(L) .EQ. L1,IFLAG .EQ. 1)) .AND. TYPE(L,3)
                                                                  (())
                                                                   .IMPLIES.
                                                                  ((LV(ENCODE,L1)) .AND. (0 .EQ.
                                                                  -ELT(LENGTH(L1),L1)+SPACE(ADDRES))
                                                                  .AND. ((0 .LE. 0 .EQ. -IFLAG+0,NO)
                                                                  .OR. (0 .LT. IFLAG-NO)) .AND. (0
                                                                  .EQ. IFLAG) .AND. ((LV(ENCODE,L))
                                                                  .OR. (0 .EQ. 1-,CDR(L)+L-L1,IFLAG))
                                                                  .AND. (TYPE(L,3)))

13
      COND LV(ENCODE,L1) .AND. ELT(LENGTH(L1),L1)=SPACE(ADDRES) .AND. ((IFLAG .EQ.
      COND 0,NO .GE. 0) .OR. (IFLAG .EQ. 1,NO .LT. 0))
             .NOT. (IFLAG .EQ. 0)
            NUMBER=-NUMBER
            CALL INSERT(ENCODE,0,IALPHN(38))
      COND LV(ENCODE,L) .AND. ((L .EQ. L1,IFLAG .EQ. 0) .OR. (ELT(1,L) .EQ.
      COND 1H-,CDR(L) .EQ. L1,IFLAG .EQ. 1)) .AND. TYPE(L,3)
                                                                  ((8))SD DCONDLV(ENCODE,L) .AND. ((L
                                                                  .EQ. L1,IFLAG .EQ. 0) .OR. (ELT(1,L)
                                                                  .EQ. 1H-,CDR(L) .EQ. L1,IFLAG .EQ.
                                                                  1)) .AND. TYPE(L,3)D DS))
                                                                   .IMPLIES.
                                                                  ((LV(ENCODE,L1)) .AND. (TYPE(L,3))
                                                                  .AND. ((LV(ENCODE,L)) .OR. (0 .EQ.
                                                                  1-,CDR(L)+L-L1,IFLAG)) .AND. (0 .NE.
                                                                  IFLAG) .AND. ((0 .LE. 0 .EQ.
                                                                  -IFLAG+0,NO) .OR. (0 .LT. IFLAG-NO))
                                                                  .AND. (0 .EQ.
                                                                  -ELT(LENGTH(L1),L1)+SPACE(ADDRES)))
```

2.13) EVCODE(NUMBER)

The argument to EVCODE is an integer and the function returns a string representing that number.

example: NUMBER=-276
result : the string "-276" is returned.

VERIFICATION:

Path 1: insufficient verification condition, since FUNCTION CALL has not been implemented into the verifier,

Path 2: Similar to path 1.

Path 3: proved.

Path 4: insufficient verification condition.

Path 5: insufficient definition for assertion.

Path 6:insufficient verification condition.

Path 7: insufficient defintion for assertion.

Path 8: Similar to above path.

Path 9: Similar to path 8.

Path 10: Similar to path 7.

Path 11: proved.

Path 12: insufficient verification conditions.

Path 13: Similar to path 7.

```
CC  LIST.PTR(F,L) .IS. EXISTS(F=NIL)'(K,L), K,1,LENGTH(L))
CC  ALL(F(X),X,I,L) .IS. (F(X). X.IN.(I..J))
CC  EXISTS(F(X),X,I,J) .IS. (F(X),I.LE.X,X.LE.J)
CC  LENGTH(L) = (IF ATL THEN 0 ELSE LENGTH(CDR(L))+1)
C   LV(LP,L) MAKES THE LIST VALUE OF LP IF L
CC  LV(LP,L) = ( F(K,(1..*)), ((1.LE.K(I), K(I).LE.50)N),
CC  SPACE(2*K(I)-1) = L(I), SPACE(2*K(I)) = 2*K(I+1) -1),
CC  I .IN. (1..*)), LP = 2*K(1) - 1)
CC  SUBLIST(J,F,L)= (IF J=K THEN NIL ELSE (CAR(L)=SPACE(J) .AND.
CC  CDR(L)=SUBLIST(SPACE(J+1),K,L)))
CCND  LV(STRIN1,L1), LV(STRIN2,L2)
      SUBROUTINE IDENT(STRIN1,STRIN2)
      COMMON/WORK/SPACE(100000),FAIL,IFJ)
      INTEGER SPACE,FAIL,STRIN1,STRIN2,STR1,STR2,CHAR1,CHAR2
C  THIS SUBROUTINE IS IDENTICAL TO THE SYMBOL FUNCTION IDENT(X,Y). IF
C  THE STRINGS STRING1 AND STRING2 ARE IDENTICAL, THIS SUBROUTINE RETURN
C  A VALUE OF #0# FOR FAIL. (IF BOTH STRINGS ARE NULL, THEY ARE
C  CONSIDERED IDENTICAL.) IF THE STRINGS ARE NOT IDENTICAL, FAIL IS
C  RETURNED AS A #1#.
      FAIL=0
      STR1=STRIN1
      STR2=STRIN2
C  CHECK IF STRING1 AND STRING2 ARE NULL OR POSSESS SAME LOCATION.
CCND  FAIL=0,LIST.PTR(STR1,L1),LIST.PTR(STR2,L2),ALL(ELT(K,L1)=ELT(K,L2),
CCND  K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))),
CCND   (LENGTH(SUBLIST(STRIN1,STR1,L1))=LENGTH(SUBLIST(STRIN1,STR2,L2)))
    1 IF(STR1-STR2) 7,6,7
    7 IF(STR1) 2,5,2
    2 IF(STR2) 3,5,3
C  PULL A CHARACTER OUT OF STR1 AND STR2
    3 CHAR1=SPACE(STR1)
      CHAR2=SPACE(STR2)
C  TEST IF CHARACTERS ARE IDENTICAL
      IF(CHAR1-CHAR2) 5,4,5
C  GET NEXT CHARACTER ADDRESS
    4 STR1=SPACE(STR1+1)
      STR2=SPACE(STR2+1)
      GO TO 1
    5 FAIL=1
CCND  (FAIL=1, EXISTS(ELT(K,L1).NE.ELT(K,L2),K,1,LENGTH(L1))) .OR.
CCND ((FAIL=0),ALL(ELT(K,L1)=ELT(K,L2),K,1,LENGTH(L1)),(LENGTH(L1)=
CCND  LENGTH(L2)))
    6 RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(STRIN1,L1) .NE. LV(STRIN2,L2)		
SUBROUTINE IDENT(STRIN1,STRIN2)		
COMMON/BLOCK/SPACE(10000),FAIL,IFOO		
INTEGER SPACE,FAIL,STRIN1,STRIN2,STR1,STR2,CHAR1,CHAR2		
FAIL=0	[FAIL]	[]
STR1=STRIN1	[STR1]	[STRIN1]
STR2=STRIN2	[STR2]	[STRIN2]
COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND.		
COND ALL(ELT(K,L1) .EQ. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1)))		
COND .AND. (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ.		
COND LENGTH(SUBLIST(STRIN1,STR2,L2)))		
1 IF(STR1-STR2 .EQ. 0)GO TO 6		
7 IF(STR1 .EQ. 0)GO TO 5		
2 IF(STR2 .EQ. 0)GO TO 5		
3 CHAR1=SPACE(STR1)	[CHAR1]	[SPACE,STR1]
CHAR2=SPACE(STR2)	[CHAR2]	[SPACE,STR2]
IF(CHAR1-CHAR2 .NE. 0)GO TO 5		
4 STR1=SPACE(STR1+1)	[STR1]	[SPACE,STR1]
STR2=SPACE(STR2+1)	[STR2]	[SPACE,STR2]
GO TO 1		
5 FAIL=1	[FAIL]	[]
COND (FAIL .EQ. 1,EXISTS(ELT(K,L1) .NE. ELT(K,L2),K,1,LENGTH(L1))) .OR.		
COND ((FAIL .EQ. 0),ALL(ELT(K,L1) .EQ.		
COND ELT(K,L2),K,1,LENGTH(L1)),(LENGTH(L1) .EQ. LENGTH(L2)))		
6 RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND (LV(STRIN1,L1) .AND. LV(STRIN2,L2)	
	SUBROUTINE ILIST(STRIN1,STRIN2)	
	COMMON/BLOCK/SPACE(10000),FAIL,IFOU	
	INTEGER SPACE,FAIL,STRIN1,STRIN2,STR1,STR2,CHAR1,CHAR2	
	FAIL=0	
	STR1=STRIN1	
	STR2=STRIN2	
	COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)	
	COND . 0. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.	
	COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))	((LV(STRIN1,L1)) .AND. (LV(STRIN2,L2))) .IMPLIES.
		((LIST.PIR(STRIN1,L1)) .AND. (LIST.PIR(STRIN2,L2)) .AND. (ALL(0 .EQ. -ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STRIN1,L1)))) .AND. (0 .EQ. -LENGTH(SUBLIST(STRIN1,STRIN1,L1))+LENGTH(SUBLIST(STRIN1,STRIN2,L2))))
2	COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)	
	COND . 0. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.	
	COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))	
	1 (STR1-STR2 .EQ. 0)	
	COND (FAIL .EQ. 1,EXISTS(ELT(K,L1) .NE. ELT(K,L2),K,1,LENGTH(L1))) .OR. ((FAIL	
	COND .EQ. 0),ALL(ELT(K,L1) .EQ. ELT(K,L2),K,1,LENGTH(L1)),(LENGTH(L1) .EQ.	
	COND LENGTH(L2)))	
		((0 .EQ. FAIL) .AND. (LIST.PIR(STR1,L1)) .AND. (LIST.PIR(STR2,L2)) .AND. (ALL(0 .EQ. -ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1)))) .AND. (0 .EQ. -LENGTH(SUBLIST(STRIN1,STR1,L1))+LENGTH(SUBLIST(STRIN1,STR2,L2))) .AND. (0 .EQ. STR1-STR2)) .IMPLIES. ((0 .EQ. -FAIL+1,EXISTS(0 .NE. -ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1))) .OR. (0 .EQ. FAIL,ALL(0 .EQ. -ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)), 0 .EQ. -LENGTH(L1)+LENGTH(L2)))
3	COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)	

```
COND .O. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.
COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))
1     .NOT. (STR1-STR2 .EQ. 0)
7     (STR1 .EQ. 0)
5     FAIL=1
COND (FAIL .EQ. 1,EXISTS(ELT(K,L1) .NE. ELT(K,L2),K,1,LENGTH(L1))) .OR. ((FAIL
COND .EQ. 0),ALL(ELT(K,L1) .EQ. ELT(K,L2),K,1,LENGTH(L1)),(LENGTH(L1) .EQ.
COND LENGTH(L2)))
```

```
((O .EQ. FAIL) .AND.
(LIST.PIR(STR1,L1)) .AND.
(LIST.PIR(STR2,L2)) .AND. (ALL(O
.EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLI
ST(STRIN1,STR1,L1)))) .AND. (O .EQ.
-LENGTH(SUBLIST(STRIN1,STR1,L1))+LENG
TH(SUBLIST(STRIN1,STR2,L2))) .AND.
(O .NE. STR1-STR2) .AND. (O .EQ.
STR1))
 .IMPLIES.
((O .EQ. -1+1,EXISTS(O .NE.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)))
 .OR. (O .EQ. 1,ALL(O .EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)),
O .EQ. -LENGTH(L1)+LENGTH(L2)))
```

```
4
COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)
COND .EQ. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.
COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))
1     .NOT. (STR1-STR2 .EQ. 0)
7     .NOT. (STR1 .EQ. 0)
2     (STR2 .EQ. 0)
5     FAIL=1
COND (FAIL .EQ. 1,EXISTS(ELT(K,L1) .NE. ELT(K,L2),K,1,LENGTH(L1))) .OR. ((FAIL
COND .EQ. 0),ALL(ELT(K,L1) .EQ. ELT(K,L2),K,1,LENGTH(L1)),(LENGTH(L1) .EQ.
COND LENGTH(L2)))
```

```
((O .EQ. FAIL) .AND.
(LIST.PIR(STR1,L1)) .AND.
(LIST.PIR(STR2,L2)) .AND. (ALL(O
.EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLI
ST(STRIN1,STR1,L1)))) .AND. (O .EQ.
-LENGTH(SUBLIST(STRIN1,STR1,L1))+LENG
TH(SUBLIST(STRIN1,STR2,L2))) .AND.
(O .NE. STR1-STR2) .AND. (O .NE.
STR1) .AND. (O .EQ. STR2))
 .IMPLIES.
((O .EQ. -1+1,EXISTS(O .NE.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)))
 .OR. (O .EQ. 1,ALL(O .EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)),
O .EQ. -LENGTH(L1)+LENGTH(L2)))
```

```
5
```

```
COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)
COND .EQ. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.
COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))
1       .NOT. (STR1-STR2 .EQ. 0)
7       .NOT. (STR1 .EQ. 0)
2       .NOT. (STR2 .EQ. 0)
3      CHAR1=SPACE(STR1)
       CHAR2=SPACE(STR2)
       (CHAR1-CHAR2 .NE. 0)
5      FAIL=1
COND (FAIL .EQ. 1,EXISTS(ELT(K,L1) .NE. ELT(K,L2),K,1,LENGTH(L1))) .OR. ((FAIL
COND .EQ. 0),ALL(ELT(K,L1) .EQ. ELT(K,L2),K,1,LENGTH(L1)),(LENGTH(L1) .EQ.
COND LENGTH(L2)))
```

```
((0 .EQ. FAIL) .AND.
(LIST.PIR(STR1,L1)) .AND.
(LIST.PIR(STR2,L2)) .AND. (ALL(0
.EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLI
ST(STRIN1,STR1,L1)))) .AND. (0 .EQ.
-LENGTH(SUBLIST(STRIN1,STR1,L1))+LENG
TH(SUBLIST(STRIN1,STR2,L2))) .AND.
(0 .NE. STR1-STR2) .AND. (0 .NE.
STR1) .AND. (0 .NE. STR2) .AND. (0
.NE. -SPACE(STR1)+SPACE(STR2)))
 .IMPLIES.
((0 .EQ. -1+1,EXISTS(0 .NE.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)))
 .OR. (0 .EQ. 1,ALL(0 .EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(L1)),
0 .EQ. -LENGTH(L1)+LENGTH(L2)))
```

```
6
COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)
COND .EQ. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.
COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))
1       .NOT. (STR1-STR2 .EQ. 0)
7       .NOT. (STR1 .EQ. 0)
2       .NOT. (STR2 .EQ. 0)
3      CHAR1=SPACE(STR1)
       CHAR2=SPACE(STR2)
        .NOT. ((CHAR1-CHAR2 .NE. 0)
4      STR1=SPACE(STR1+1)
       STR2=SPACE(STR2+1)
COND FAIL=0 .AND. LIST.PIR(STR1,L1) .AND. LIST.PIR(STR2,L2) .AND. ALL(ELT(K,L1)
COND .EQ. ELT(K,L2),K,1,LENGTH(SUBLIST(STRIN1,STR1,L1))) .AND.
COND (LENGTH(SUBLIST(STRIN1,STR1,L1)) .EQ. LENGTH(SUBLIST(STRIN1,STR2,L2)))
```

```
((0 .EQ. FAIL) .AND.
(LIST.PIR(STR1,L1)) .AND.
(LIST.PIR(STR2,L2)) .AND. (ALL(0
.EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLI
ST(STRIN1,STR1,L1)))) .AND. (0 .EQ.
-LENGTH(SUBLIST(STRIN1,STR1,L1))+LENG
TH(SUBLIST(STRIN1,STR2,L2))) .AND.
(0 .NE. STR1-STR2) .AND. (0 .NE.
```

```
STR1) .AND. (0 .NE. STR2) .AND. (0
.EQ. -SPACE(STR1)+SPACE(STR2)))
 .IMPLIES.
((LIST.PIR(SPACE(1+STR1),L1)) .AND.
(LIST.PIR(SPACE(1+STR2),L2)) .AND.
(0 .EQ.
-LENGTH(SUBLIST(STRIN1,SPACE(1+STR1),
L1))+LENGTH(SUBLIST(STRIN1,SPACE(1+ST
R2),L2))) .AND. (ALL(0 .EQ.
-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLI
ST(STRIN1,SPACE(1+STR1),L1)))))
```

2.14) IDENT(STRIN1,STRIN2)

This subroutine checks to see whether or not STRIN1 is a string that is identical to the string STRIN2. If so, FAIL is set to 0 and if not, FAIL is set to 1. STRIN1 and STRIN2 may or may not be pointers to the same string.

example: STRIN1 = "SAME" STRIN2 = "SAME"
result : FAIL = 0

The verification conditions of the paths are:

Path 1 With the properties that LENGTH(SUBLIST(J,J,L))=0 and ALL(F(X),X,1,0)=.TRUE., the verification condition is simplified to (LV(STRIN1,L1).AND. LV(STRIN2,L2)) implies (LIST.PIR(STRIN1,L1).AND. LIST.PIR(STRIN2,L2)).

Path 2 The verification condition of this path is
(ALL(0.EQ.-ELT(K,L1)+ELT(K,L2),K,1,LENGTH(SUBLIST(
STRIN1,STR1,L1).AND.(0.EQ.-LENGTH(SUBLIST(STRIN1,
STR1,L1))+LENGTH(SUBLIST(STRIN2,STR2,L2))).AND.
(0.EQ.STR1-STR2))
implies
(0.EQ.FAIL,ALL(0.EQ.-ELT(K,L1)+ELT(K,L2),K,1,
LENGTH(L1)),(0.EQ.-LENGTH(L1)+LENGTH(L2)))

Path 4 The verification condition of this path is the same as Path 3 except (0.EQ.STR1) in the hypothesis is replaced by (0.EQ.STR2).

Path 3 The verification condition of this path is
(LIST.PIR(STR1,L1).AND.LIST.PIR(STR2,L2).AND.(0.EQ.
-LENGTH(SUBLIST(STRIN1,STR1,L1))+LENGTH(SUBLIST(
STRIN2,STR2,L2))).AND.(0.NE.STR1-STR2).AND.(0.EQ.STR1))
implies
(FAIL.EQ.1,EXISTS(ELT(K,L1).NE.ELT(K,L2),K,1,LENGTH(L1)).

Path 5 The verification condtion of this path is
(the whole hypothesis of the path)
implies
(FAIL.EQ.1,EXISTS(ELT(K,L1).NE.ELT(K,L2),K,1,LENGTH(L1)).

Path 6 The verification condition of the path is
(the whole hypothesis of the path)
implies
(the whole conclusian of the path)
The verification of this path depends on the property that LENGTH(SUBLIST(STRIN1,SPACE(STR1+1),L1))=LENGTH(SUBLIST(STRIN1,STR1,L1))+1, for (SPACE(STR1+1).NE.0).

```
      SUBROUTINE INITIA
CC SPACE(N)
      COMMON/BLOCK/SPACE(10000),F,FREE
CC SPACE(N)
      INTEGER SPACE,FREE,F
C INITIALIZE FREE POINTER
      FREE = 1
      I=2
C SET UP SINGLY LINKED LIST
CC I=2,N-2,2
COND   I .GE. 2,I .LE. 9998,FREE=1
  100 SPACE(I)=I+1
      I=I+2
COND   SPACE(I-2)=I-1,I-2 .LE. 9998,FREE=1,I-2 .GE. 2
      IF(I .GT. 9998) GO TO 101
      GO TO 100
C SET END POINTER
CC SPACE(N) = 0
  101 SPACE(10000)=0
COND   SPACE(I-2)=I-1,SPACE(10000)=0,I-2 .LE. 9998,I-2 .GE. 2,FREE=1
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
SUBROUTINE INITIA		
COMMON/BLOCK/SPACE(10000),F,FREE		
INTEGER SPACE,FREE,F		
FREE=1	[FREE]	[FREE]
I=2	[I]	[I]
COND I .GE. 2 .AND. I .LE. 9998 .AND. FREE=1		
100 SPACE(I)=I+1	[SPACE]	[SPACE,I]
I=I+2	[I]	[I]
COND SPACE(I-2)=I-1 .AND. I-2 .LE. 9998 .AND. FREE=1 .AND. I-2 .GE. 2		
IF(I .GT. 9998)GO TO 101		
GO TO 100		
101 SPACE(10000)=0	[SPACE]	[SPACE]
COND SPACE(I-2)=I-1 .AND. SPACE(10000)=0 .AND. I-2 .LE. 9998 .AND. I-2		
COND .GE. 2 .AND. FREE=1		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND .TRUE. SUBROUTINE INITIA COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,FREE,F FREE=1 I=2 COND I .GE. 2 .AND. I .LE. 9998 .AND. FREE=1	0 .LE. 9998
2	COND I .GE. 2 .AND. I .LE. 9998 .AND. FREE=1 100 SPACE(I)=I+1 I=I+2 COND SPACE(I-2)=I-1 .AND. I-2 .LE. 9998 .AND. FREE=1 .AND. I-2 .GE. 2	NONE
3	COND SPACE(I-2)=I-1 .AND. I-2 .LE. 9998 .AND. FREE=1 .AND. I-2 .GE. 2 (I .GT. 9998) 101 SPACE(10000)=I COND SPACE(I-2)=I-1 .AND. SPACE(10000)=0 .AND. I-2 .LE. 9998 .AND. I-2 .GE. 2 COND .AND. FREE=1	NONE
4	COND SPACE(I-2)=I-1 .AND. I-2 .LE. 9998 .AND. FREE=1 .AND. I-2 .GE. 2 .NOT. (I .GT. 9998) COND I .GE. 2 .AND. I .LE. 9998 .AND. FREE=1	NONE
3		
2		
1		

2.15) INITIA

This subroutine initializes SPACE to be a string of available space and sets FREE to point to the first cell of the string.

VERIFICATION:

PATH 1: Obvious.

PATH 2: proved.

PATH 3: proved.

PATH 4: proved.

```
CC   LENGTH(L) = (IF [illegible] THEN [illegible] ELSE LENGTH(CDR(L))+1)
C    LV(LP,L) MEANS THE LIST VALUE OF LP IS L
CC   LV(LP,L) = ([illegible],(1..I)), (I1.LE.N(I), N(I).LE.5000,
CC   SPACE(2*N(I)-1) = [illegible] SPACE(2*N(I)) - 2*N(I+1) -1).
CC   I .IN. (1..N), LP = 2*N(1) - 1)
CC   [illegible](0,L)=L) [illegible] CDR(I,L)=[illegible](I-1,L))
CC   ALL(P(X),X,I,J) .IFF. (P(X), X.IN.(I..J))
CC   EXISTS(P(X),X,I,J) .IFF. (P(X),I.LE.X,X.LE.J))
CC   LIST.PIR(N,L) .IFF. EXISTS(N=A[illegible](I,L),I,1,LENGTH(L))
C
COND    LV(STRING,L), ELEM.GE.0
        SUBROUTINE I[illegible]T(STRING,ELEM,DATA)
        COMMON/[illegible]/[illegible](1000),IF[illegible],IF[illegible]
        INTEGER SPACE,STRING,ELEM,DATA

C WALK THRU LIST TO ELEM-1 ELEMENT
        INDEX = STRING
COND    LV(STRING,L), INDEX=STRING, ELEM.GE.0
        IF(ELEM-1) 290,280,200
200     K = ELEM - 1
        J = 1
COND    ELEM.GT.1, K=ELEM-1, LV(STRING,L), LIST.PIR(INDEX,L),
COND    EXISTS(ELT(J,L)=SPACE(INDEX),J,1,K)
250     INDEX = SPACE(INDEX + 1)
        J = J + 1
        IF (J .LE. K) GO TO 250
C INSERT CELL
COND    ELEM.NE.0, LIST.PIR(INDEX,L), LV(STRING,L),
COND    EXISTS(ELT(ELEM,L)=SPACE(INDEX),ELEM,1,LENGTH(L))
280     NEW = NEWCEL(0)
        SPACE(NEW+1) = SPACE(INDEX+1)
        SPACE(INDEX+1) = NEW
        SPACE(NEW) = DATA
COND     ELEM .NE. 0,LV(STRING,LL),ALL(ELT(M,LL)=ELT(M,L),M,1,
COND    ELEM),(ELT(ELEM+1,LL)=DATA),(CDRN(ELEM+1,LL)=CDRN(ELEM,L))
        RETURN
290     NEW = NEWCEL(0)
        SPACE(NEW+1) = STRING
        SPACE(NEW) = DATA
        STRING = NEW
COND     (LV(STRING,LL),(CAR(LL)=DATA),(CDR(LL)=L),(ELEM=0))
        RETURN
        END
--------------------------
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(STRING,L) .AND. ELEM .GE. 0		
SUBROUTINE INSERT(STRING,ELEM,DATA)		
COMMON/BLOCK/SPACE(10000),IFQ0,IFJJ		
INTEGER SPACE,STRING,ELEM,DATA		
INDEX=STRING	[INDEX]	[STRING]
COND LV(STRING,L) .AND. INDEX=STRING .AND. ELEM .GE. 0		
IF(ELEM-1 .LT. 0)GO TO 290		
IF(ELEM-1 .EQ. 0)GO TO 280		
200 K=ELEM-1	[K]	[ELEM]
J=1	[J]	[]
COND ELEM .GT. 1 .AND. K=ELEM-1 .AND. LV(STRING,L) .AND.		
COND LIST.PTR(INDEX,L) .AND. EXISTS(ELT(J,L) .EQ. SPACE(INDEX),J,1,K)		
250 INDEX=SPACE(INDEX+1)	[INDEX]	[SPACE,INDEX]
J=J+1	[J]	[]
IF(J .LE. K)GO TO 250		
COND ELEM .NE. 0 .AND. LIST.PTR(INDEX,L) .AND. LV(STRING,L) .AND.		
COND EXISTS(ELT(ELEM,L) .EQ. SPACE(INDEX),ELEM,1,LENGTH(L))		
280 NEW=NEWCEL(0)	[NEW]	[NEWCEL]
SPACE(NEW+1)=SPACE(INDEX+1)	[SPACE]	[SPACE,NEW,INDEX]
SPACE(INDEX+1)=NEW	[SPACE]	[SPACE,INDEX,NEW]
SPACE(NEW)=DATA	[SPACE]	[SPACE,NEW,DATA]
COND ELEM .NE. 0 .AND. LV(STRING,LL) .AND. ALL(ELT(M,LL) .EQ.		
COND ELT(M,L),M,1,ELEM) .AND. (ELT(ELEM+1,LL) .EQ. DATA) .AND.		
COND (CDR(ELEM+1,LL) .EQ. CDR(ELEM,L))		
RETURN		
290 NEW=NEWCEL(0)	[NEW]	[NEWCEL]
SPACE(NEW+1)=STRING	[SPACE]	[SPACE,NEW,STRING]
SPACE(NEW)=DATA	[SPACE]	[SPACE,NEW,DATA]
STRING=NEW	[STRING]	[NEW]
COND (LV(STRING,LL),(CAR(LL) .EQ. DATA),(CDR(LL) .EQ. L),(ELEM .EQ. 0))		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS (IF ANY) TO BE VISITED
1	COND LV(STRING,L) .AND. ELEM .GE. 0 SUBROUTINE INSERT(STRING,ELEM,DATA) COMMON/BLOCK/SPACE(10000),IFOO,IFOO1 INTEGER SPACE,STRING,ELEM,DATA INDEX=STRING COND LV(STRING,L) .AND. INDEX=STRING .AND. ELEM .GE. 0	((LV(STRING,L)) .AND. (J .LE. LL+N)) .IMPLIES. ((LV(STRING,L)))
2	COND LV(STRING,L) .AND. INDEX=STRING .AND. ELEM .GE. 0 (ELEM-1 .LT. 0) 290 NEW=GETCELL(0) SPACE(NEW+1)=STRING SPACE(NEW)=DATA STRING=NEW COND (LV(STRING,LL)),(CAR(LL) .EQ. DATA),(CDR(LL) .EQ. L),(ELEM .EQ. 0))	(((LV(STRING,L)) .AND. (J .E. -(INDEX+STRING) .AND. (J .LE. ELEM) .AND. (J .LT. 1-FLE=1)) .IMPLIES. ((LV(NEWCELL(J),LL),J .EJ. -CAR(LL)+DATA,J .EJ. -CDR(LL)+L,J .EJ. ELEM)))
3	COND LV(STRING,L) .AND. INDEX=STRING .AND. ELEM .GE. 0 .NOT. (ELEM-1 .LT. 0) (ELEM-1 .EQ. 0) COND ELEM .NE. 0 .AND. LIST.PTR(INDEX,L) .AND. LV(STRING,L) .AND. COND EXISTS(ELT(ELEM,L) .EQ. SPACE(INDEX),ELEM,1,LENGTH(L))	((LV(STRING,L)) .AND. (J .EJ. -(INDEX+STRING) .AND. (J .LE. .ELEM) .AND. (J .LE. -1+ELEM) .AND. (J .E. -1+ELEM))) .IMPLIES. ((LIST.PTR(INDEX,L)) .AND. (LV(STRING,L)) .AND. (EXISTS(J .EJ. -ELT(ELEM,L)+SPACE(INDEX),ELEM,1,LENG TH(L))) .AND. (J .NE. -ELEM))
4	COND LV(STRING,L) .AND. INDEX=STRING .AND. ELEM .GE. 0 .NOT. (ELEM-1 .LT. 0) .NOT. (ELEM-1 .EQ. 0) 200 K=ELEM-1	

```
      J=1
COND ELEM .GT. 1 .AND. K=ELEM-1 .AND. LV(STRING,L) .AND. LIST.PIR(INDEX,L)
COND .AND. EXISTS(ELT(J,L) .EQ. SPACE(INDEX),J,1,K)
```

```
((LV(STRING,L)) .AND. () .E).
-INDEX+STRING) .AND. () .LE. ELEM)
.AND. (0 .LE. -1+ELEM) .AND. (0 .NE.
-1+ELEM))
 .IMPLIES.
((LV(STRING,L)) .AND.
(LIST.PIR(INDEX,L)) .AND. (EXISTS()
.E).
-ELT(1,L)+SPACE(INDEX),1,1,-1+ELEM))
.AND. (J .LT. -1+ELEM))
```

```
COND ELEM .GT. 1 .AND. K=ELEM-1 .AND. LV(STRING,L) .AND. LIST.PIR(INDEX,L)
COND .AND. EXISTS(ELT(J,L) .EQ. SPACE(INDEX),J,1,K)
250   INDEX=SPACE(INDEX+1)
      J=J+1
      (J .LE. K)
COND ELEM .GT. 1 .AND. K=ELEM-1 .AND. LV(STRING,L) .AND. LIST.PIR(INDEX,L)
COND .AND. EXISTS(ELT(J,L) .EQ. SPACE(INDEX),J,1,K)
```

```
() .LT. -1+ELEM) .AND. () .E).
-1+ELEM-K) .AND. (LV(STRING,L))
.AND. (LIST.PIR(INDEX,L)) .AND.
(EXISTS() .E).
-ELT(J,L)+SPACE(INDEX),J,1,K)) .AND.
() .LE. -1-J+K))
 .IMPLIES.
((LV(STRING,L)) .AND.
(LIST.PIR(SPACE(1+INDEX),L)) .AND.
(EXISTS() .E).
-ELT(1+J,L)+SPACE(SPACE(1+INDEX)),1+J
,1,K)))
```

```
COND ELEM .GT. 1 .AND. K=ELEM-1 .AND. LV(STRING,L) .AND. LIST.PIR(INDEX,L)
COND .AND. EXISTS(ELT(J,L) .EQ. SPACE(INDEX),J,1,K)
250   INDEX=SPACE(INDEX+1)
      J=J+1
       .NOT. (J .LE. K)
COND ELEM .E. J .AND. LIST.PIR(INDEX,L) .AND. LV(STRING,L) .AND.
COND EXISTS(ELT(ELEM,L) .EQ. SPACE(INDEX),ELEM,1,LENGTH(L))
```

```
() .LT. -1+ELEM) .AND. () .E).
-1+ELEM-K) .AND. (LV(STRING,L))
.AND. (LIST.PIR(INDEX,L)) .AND.
(EXISTS() .E).
-ELT(J,L)+SPACE(INDEX),J,1,K)) .AND.
() .LT. 1+J-K))
 .IMPLIES.
((LIST.PIR(SPACE(1+INDEX),L)) .AND.
(LV(STRING,L)) .AND. (EXISTS() .E).
-ELT(ELEM,L)+SPACE(SPACE(1+INDEX)),EL
EM,1,LENGTH(L))) .AND. () .E.
```

```
                                                                    -ELEM))

7                    !
COND ELEM .NE. 0 .AND. LIST.PTR(INDEX,L) .AND. LV(STRING,L) .AND.
COND EXISTS(-LT(ELEM,L) .EQ. SPACE(INDEX),ELEM,1,LENGTH(L))
240    .EQ= PROCEL())
       SPACE(INDEX+1)=SPACE(INDEX+1)
       SPACE(INDEX+1)=NEW
       SPACE(NEW)=DATA
COND ELEM .NE. 0 .AND. LV(STRING,LL) .AND. ALL(ELT(M,LL) .EQ.
COND ELT(M,L),M,1,ELEM) .AND. (ELT(ELEM+1,LL) .EQ. DATA) .AND. (CORN(ELEM+1,LL)
COND .EQ. CO N((ELEM,L))
                                                                    ((0 .EQ. -ELEM)) .AND.
                                                                    (LIST.PTR((INDEX,L)) .AND.
                                                                    (LV(STRING,L)) .AND. (EXISTS() .EQ.
                                                                    -ELT(ELEM,L)+SPACE((INDEX),ELEM,1,LENG
                                                                    TH(L))))
                                                                     .IMPLIES.
                                                                    (((LV(STRING,LL)) .AND. () .EQ.
                                                                    -CORN((ELEM+1),LL)+CO)N.((ELEM,L)) .AND.
                                                                    (0 .EQ. DATA=ELT((ELEM+1),LL)) .AND.
                                                                    (ALL(0 .EQ).
                                                                    -ELT(M,LL)+ELT(M,L),M,1,"ELEM)) .AND.
                                                                    () .NE. -ELEM)))
```

2.16) INSERT(STRING,ELEM,DATA)

This inserts the character DATA into the (ELEM+1)-th position of STRING.

example: STRING = "INSERT" ELEM = 1 DATA = 1HN
result : STRING = "INSERT"

The verification of the paths are:

Path 1 Verification condition is obvious.

Path 2 Verification of this path is not clear. There seems to be some insufficiency in the concluding assertion.

Path 3 By mathematical manipulation of the hypothesis, it can easily be shown that ELEM=1. Thus the verification condition is obvious.

Path 4 By mathematical manipulation of the hypothesis, it can easily be shown that ELEM.GT.1. Thus the verification condition is obvious.

Path 5 It seems that the statement (ELEM.LT.LENGTH(L)) must be added to the initial and final assertions of this path for the verification condition to be sufficient.

Path 6 With the hypothesis implying (J=ELEM), the verification condition is obvious.

Path 7 Some refinement of the mathematical function CDRN(I,L) is needed before the verification condition is sufficient.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
COND   LV(NAME,L)
      INTEGER FUNCTION ISIZE(NAME)
      COMMON/BLOCK/SPACE(10000),IQQ,IQQQ
      INTEGER SPACE
      ISIZE = 0
C TEST FOR NULL STRING
      IF(NAME) 20,10,20
COND   ISIZE=LENGTH(L)
 10   RETURN
 20   INDEX = NAME
COND   LV(INDEX,CDRN(ISIZE,L))
 30   ISIZE = ISIZE + 1
C SET ITEST TO NEXT POINTER
      ITEST = SPACE(INDEX+1)
C TEST FOR END OF STRING
      IF(ITEST) 35,40,35
 35   INDEX = ITEST
      GO TO 30
COND   ISIZE=LENGTH(L)
 40   RETURN
      END
-----------------------
```

```
        PROGRAM [illegible] VERIFIED                                    EFFECTIVE [illegible]    [illegible]
        ---------------------                                           ---------------

 COND LV(NAME,L)
      INTEGER FUNCTION ISIZE(NAME)
      COMMON/BLOCK/SPACE(10000),IQQ,IQQQ
      INTEGER SPACE
      ISIZE=0                                                          [ISIZE]          []
      IF(NAME .NE. 0)GO TO 20
 COND ISIZE=LENGTH(L)
 10   RETURN
 20   INDEX=NAME                                                       [INDEX]          [NAME]
 COND LV(INDEX,CDRN(ISIZE,L))
 30   ISIZE=ISIZE+1                                                    [ISIZE]          [ISIZE]
      ITEST=SPACE(INDEX+1)                                             [ITEST]          [SPACE,INDEX]
      IF(ITEST .EQ. 0)GO TO 40
 35   INDEX=ITEST                                                      [INDEX]          [ITEST]
      GO TO 30
 COND ISIZE=LENGTH(L)
 40   RETURN
      END
```

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(NAME,L) INT GLR FUNCTION ISIZE(NAME) COMMON/BLOCK/SPACE(10000),IQQ,IQQQ INTEGER SPACE ISIZE=0 (NAME .NE. 0) 20 INDEX=NAME COND LV(INDEX,CDRN(ISIZE,L))	((LV(NAME,L)) .AND. (0 .NE. -NAME)) .IMPLIES. ((LV(NAME,CDRN(0,L))))
2	COND LV(NAME,L) INTEGER FUNCTION ISIZE(NAME) COMMON/BLOCK/SPACE(10000),IQQ,IQQQ INTEGER SPACE ISIZE=0 .NOT. (NAME .NE. 0) COND ISIZE=LENGTH(L)	((LV(NAME,L)) .AND. (0 .EQ. -NAME)) .IMPLIES. ((0 .EQ. LENGTH(L)))
3	COND LV(INDEX,CDRN(ISIZE,L)) 30 ISIZE=ISIZE+1 ITEST=SPACE(INDEX+1) (ITEST .EQ. 0) COND ISIZE=LENGTH(L)	((LV(INDEX,CDRN(ISIZE,L))) .AND. (0 .EQ. SPACE(1+INDEX))) .IMPLIES. ((0 .EQ. -1-ISIZE+LENGTH(L)))
4	COND LV(INDEX,CDRN(ISIZE,L)) 30 ISIZE=ISIZE+1 ITEST=SPACE(INDEX+1) .NOT. (ITEST .EQ. 0) 35 INDEX=ITEST COND LV(INDEX,CDRN(ISIZE,L))	((LV(INDEX,CDRN(ISIZE,L))) .AND. (0 .NE. SPACE(1+INDEX))) .IMPLIES. ((LV(SPACE(1+INDEX),CDRN(1+ISIZE,L))))

```
CC      LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC      SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
COND     LV(LINE,L1),MLENGT .GT. MSTART, MSTART .GT. 0
        SUBROUTINE JOUTPU(LINE,MSTART,MLENGT)
        COMMON/ALPHNU/Q(63)/CON/ICOND
        INTEGER SPACE,Q
        COMMON/ARRAYS/IVECT(130)/BLOCK/SPACE(10000),IFAIL,IFREE
        IF(LINE .EQ.0) GO TO 9
        CALL COPPY(K,LINE,0)
        NFLAG=0
        IF(SPACE(K).EQ.Q(43))GO TO 8
COND     MLENGT .GT. MSTART, MSTART .GT. 0
        CALL SUBST(K)
        IF(MSTART.EQ.1.AND.MLENGT.EQ.130)GO TO 7
        CALL STRING(LINE,ICOND,I,IJ,1)
        IF(IFAIL.EQ.0)NFLAG=1
    7   I=1
COND     I .LE. MSTART .OR. MSTART .LT. 1, MLENGT .GT. MSTART,
COND     MSTART .GT. 0
    1   IVECT(I)=Q(45)
        I=I+1
        IF(I.LE. MSTART) GO TO 1
COND     I .GT. MSTART, MLENGT .GT. MSTART, MSTART .GT. 0
        IVECT(MSTART+4)=Q(45)
        KKK=MSTART+MLENGT-1
        NSTART=MSTART
    5   IFLAG=0
        I=NSTART
COND     I .LE. KKK, MLENGT .GT. MSTART, MSTART .GT. 0
   22   IF(I.EQ.MSTART+4.AND.NFLAG.EQ.1)GO TO 2
        IVECT(I)=SPACE(K)
        IF(IVECT(I).EQ.Q(45)) IFLAG=1
        CALL DELETA(K,0)
        IF(K.EQ.0)GO TO 3
    2   I=I+1
        IF(I .LT. KKK) GO TO 22
        I=KKK
        IF(IFLAG.EQ.0)GO TO 3
        IJ=I+1
        III=MLENGT-1
        II=1
COND     II .LE. III, I .LE. KKK, MLENGT .GT. MSTART, MSTART .GT. 0
    6   IF(IVECT(IJ-II).EQ.Q(45))GO TO 3
        CALL INSERT(K,0,IVECT(IJ-II))
        I=I-1
        II=II+1
        IF(II .LT. III) GO TO 6
    3   WRITE(6,4)(IVECT(J),J=1,I)
    4   FORMAT(1X,130A1)
COND     I .LE. KKK,
COND     ALL((OUTPUT=IVECT(JJ)),JJ,1,I)
        IF(NFLAG.EQ.1)NSTART=MSTART+5
        IF(K.GT.0)GO TO5
COND     K .LE. 0, ALL((OUTPUT=IVECT(JJ)), JJ,1,I)
        RETURN
```

```
8       CALL DELETA(K,0)
        GO TO 7
COND     LV(LINE,L1)=NIL
     9  RETURN
        END
```

2.17) ISIZE(NAME)

The argument NAME is a string. The result returned is the length of that string. This is similar to the ISIZE function in SNOBOL.

example: NAME = "LONGSTRING"
result : The integer 10 is returned.

VERIFICATION:

PATH 1: Since CDRN(0,L)=L, hence the conclusion is correct.

PATH 2: Since NAME=0, L=NIL and its length is 0.

PATH 3: Since LV(INDEX,CDRN(ISIZE,L)) and (0=SPACE(1+INDEX)), hence LENGTH(L)=ISIZE+1.

PATH 4: Similar to the above path except SPACE(INDEX)≠0. Hence LV(SPACE(1+INDEX),CDRN(1+ISIZE,L)) is true.

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(LINE,L1) .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0		
SUBROUTINE OUTPUT(LINE,MSTART,MLENGT)		
COMMON/ALPHNU/Q(63)/CON/ICOND		
INTEGER SPACE,Q		
COMMON/ARRAYS/IVECT(130)/BLOCK/SPACE(10000),IFAIL,IFREE		
IF(LINE .EQ. 0)GO TO 9		
CALL COPY(K,LINE,0)		
NFLAG=0	[NFLAG]	[]
IF(SPACE(K) .EQ. Q(43))GO TO 8		
COND MLENGT .GT. MSTART .AND. MSTART .GT. 0		
CALL SUBST(K)		
IF(MSTART .EQ. 1 .AND. MLENGT .EQ. 130)GO TO 7		
CALL STRING(LINE,ICOND,I,IJ,1)		
IF(IFAIL .EQ. 0)GO TO 23		
GO TO 24		
23 NFLAG=1	[NFLAG]	[]
24 CONTINUE		
7 I=1	[I]	[]
COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND.		
COND MSTART .GT. 0		
1 IVECT(I)=Q(45)	[IVECT]	[IVECT,I,Q]
I=I+1	[I]	[I]
IF(I .LE. MSTART)GO TO 1		
COND I .GT. MSTART .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0		
IVECT(MSTART+4)=Q(45)	[IVECT]	[IVECT,MSTART,Q]
KKK=MSTART+MLENGT-1	[KKK]	[MSTART,MLENGT]
NSTART=MSTART	[NSTART]	[MSTART]
5 IFLAG=0	[IFLAG]	[]
I=NSTART	[I]	[NSTART]
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0		
22 IF(I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)GO TO 2		
IVECT(I)=SPACE(K)	[IVECT]	[IVECT,I,SPACE,K]
IF(IVECT(I) .EQ. Q(45))GO TO 25		
GO TO 26		
25 IFLAG=1	[IFLAG]	[]
26 CONTINUE		
CALL DELTA(K,0)		
IF(K .EQ. 0)GO TO 3		
2 I=I+1	[I]	[I]
IF(I .LE. KKK)GO TO 22		
I=KKK	[I]	[KKK]
IF(IFLAG .EQ. 0)GO TO 3		
IJ=I+1	[IJ]	[I]
III=MLENGT-1	[III]	[MLENGT]
II=1	[II]	[]
COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART		
COND .GT. 0		
6 IF(IVECT(IJ-II) .EQ. Q(45))GO TO 3		
CALL INSERT(K,0,IVECT(IJ-II))		

```
      I=I-1                                                    [I]          [I]
      II=II+1                                                  [II]         [II]
      IF(II .LE. III)GO TO 6
3     WRITE(6,4)(IVECT(J),J=1,I)
4     FORMAT(1X,130A1)
COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
      IF(NFLAG .EQ. 1)GO TO 27
      GO TO 28
27    NSTART=MSTART+5                                          [NSTART]     [MSTART]
28    CONTINUE
      IF(K .GT. 0)GO TO 5
COND K .LE. 0 .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
      RETURN
8     CALL DELETA(K,0)
      GO TO 7
COND LV(LINE,L1)=NIL
9     RETURN
      END
```

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(LINE,L1) .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0 SUBROUTINE JOUTPU(LINE,MSTART,MLENGT) COMMON/ALPHNU/Q(63)/CON/ICOND INTEGER SPACE,Q COMMON/ARRAYS/IVECT(130)/BLOCK/SPACE(10000),IFAIL,IFREE (LINE .EQ. 0) COND LV(LINE,L1)=NIL	((LV(LINE,L1)) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART) .AND. (0 .EQ. LINE)) .IMPLIES. ((0 .EQ. -LV(LINE,L1)+NIL))
2	COND LV(LINE,L1) .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0 SUBROUTINE JOUTPU(LINE,MSTART,MLENGT) COMMON/ALPHNU/Q(63)/CON/ICOND INTEGER SPACE,Q COMMON/ARRAYS/IVECT(130)/BLOCK/SPACE(10000),IFAIL,IFREE .NOT. (LINE .EQ. 0) CALL COPPY(K,LINE,0) NFLAG=0 (SPACE(K) .EQ. Q(43)) 8 CALL DELETA(K,0) 7 I=1 COND I .LT. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART COND .GT. 0	((LV(LINE,L1)) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART) .AND. (0 .NE. LINE) .AND. (0 .EQ. Q(43)-SPACE(K))) .IMPLIES. ((0 .LE. -1+MSTART) .OR. ((0 .LT. 1-MSTART) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART)))
3	COND LV(LINE,L1) .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0 SUBROUTINE JOUTPU(LINE,MSTART,MLENGT) COMMON/ALPHNU/Q(63)/CON/ICOND INTEGER SPACE,Q COMMON/ARRAYS/IVECT(130)/BLOCK/SPACE(10000),IFAIL,IFREE .NOT. (LINE .EQ. 0) CALL COPPY(K,LINE,0) NFLAG=0 .NOT. (SPACE(K) .EQ. Q(43)) COND MLENGT .GT. MSTART .AND. MSTART .GT. 0	

NONE 1

4

```
      COND MLENGT .GT. MSTART .AND. MSTART .GT. 0
           CALL SUBST(K)
           (MSTART .EQ. 1 .AND. MLENGT .EQ. 130)                                    2
      7    I=1
      COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART
      COND .GT. 0
                                                                                    ((0 .LT. MLENGT-MSTART) .AND. (0
                                                                                    .LT. MSTART) .AND. ((0 .LQ.
                                                                                    1-MSTART) .AND. (0 .EQ. 130-MLENGT)))   3
                                                                                     .IMPLIES.
                                                                                    ((0 .LE. -1+MSTART) .OR. ((0 .LT.
                                                                                    1-MSTART) .AND. (0 .LT.
                                                                                    MLENGT-MSTART) .AND. (0 .LT.
                                                                                    MSTART)))
```

5

```
      COND MLENGT .GT. MSTART .AND. MSTART .GT. 0
           CALL SUBST(K)
            .NOT. (MSTART .EQ. 1 .AND. MLENGT .EQ. 130)
           CALL STRING(LINE,ICOND,I,IJ,1)
           (IFAIL .EQ. 0)
      23   NFLAG=1
      24   CONTINUE
      7    I=1
      COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART
      COND .GT. 0
                                                                                    ((0 .LT. MLENGT-MSTART) .AND. (0
                                                                                    .LT. MSTART) .AND. ( .NOT. ((0 .EQ.
                                                                                    1-MSTART) .AND. (0 .EQ.
                                                                                    130-MLENGT))) .AND. (0 .EQ. IFAIL))
                                                                                     .IMPLIES.
                                                                                    ((0 .LE. -1+MSTART) .OR. ((0 .LT.
                                                                                    1-MSTART) .AND. (0 .LT.
                                                                                    MLENGT-MSTART) .AND. (0 .LT.
                                                                                    MSTART)))
```

6

```
      COND MLENGT .GT. MSTART .AND. MSTART .GT. 0
           CALL SUBST(K)
            .NOT. (MSTART .EQ. 1 .AND. MLENGT .EQ. 130)
           CALL STRING(LINE,ICOND,I,IJ,1)
            .NOT. (IFAIL .EQ. 0)
      24   CONTINUE
      7    I=1
      COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART
      COND .GT. 0
                                                                                    ((0 .LT. MLENGT-MSTART) .AND. (0
                                                                                    .LT. MSTART) .AND. ( .NOT. ((0 .EQ.
                                                                                    1-MSTART) .AND. (0 .EQ.
                                                                                    130-MLENGT))) .AND. (0 .NE. IFAIL))
```

```
                                                                  .IMPLIES.
                                                                  ((0 .LE. -1+MSTART) .OR. ((0 .LT.
                                                                  1-MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LT.
                                                                  MSTART)))

7
      COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART
      COND .GT. 0
      1     IVECT(I)=Q(45)
            I=I+1
            (I .LE. MSTART)
      COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART
      COND .GT. 0
                                                                  (((0 .LE. -I+MSTART) .OR. ((0 .LT.
                                                                  1-MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LT.
                                                                  MSTART))) .AND. (0 .LE. -1-I+MSTART))
                                                                  .IMPLIES.
                                                                  ((0 .LE. -1-I+MSTART) .OR. ((0 .LT.
                                                                  1-MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LT.
                                                                  MSTART))).

8
      COND I .LE. MSTART .OR. MSTART .LT. 1 .AND. MLENGT .GT. MSTART .AND. MSTART
      COND .GT. 0
      1     IVECT(I)=Q(45)
            I=I+1
            .NOT. (I .LE. MSTART)
      COND I .GT. MSTART .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
                                                                  (((0 .LE. -I+MSTART) .OR. ((0 .LT.
                                                                  1-MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LT.
                                                                  MSTART))) .AND. (0 .LT. 1+I-MSTART))
                                                                  .IMPLIES.
                                                                  ((0 .LT. MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART))

9
      COND I .GT. MSTART .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
            IVECT(MSTART+4)=Q(45)
            KKK=MSTART+MLENGT-1
            MSTART=MSTART
      5     IFLAG=0
            I=MSTART
      COND I .LT. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
                                                                  ((0 .LT. I-MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LT. MSTART))
                                                                  .IMPLIES.
                                                                  ((0 .LE. -1+MLENGT))
```

10

```
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22    (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
2     I=I+1
      (I .LE. KKK)
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
```

NONE

11

```
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22    (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
2     I=I+1
       .NOT. (I .LE. KKK)
      I=KKK
      (IFLAG .EQ. 0)
3     WRITE(6,4)(IVECT(J),J=1,I)
4     FORMAT(1X,130A1)
COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
```

((0 .LE. -I+KKK) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART) .AND. ((0 .EQ. 4-I+MSTART) .AND. (0 .EQ. 1-NFLAG)) .AND. (0 .LT. 1+I-KKK) .AND. (0 .EQ. IFLAG)) .IMPLIES. ((ALL(0 .EQ. IVECT(JJ)-OUTPUT,JJ,1,KKK)))

12

```
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22    (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
2     I=I+1
       .NOT. (I .LE. KKK)
      I=KKK
       .NOT. (IFLAG .EQ. 0)
      IJ=I+1
      III=MLENGT-1
      II=1
COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
```

((0 .LE. -I+KKK) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART) .AND. ((0 .EQ. 4-I+MSTART) .AND. (0 .EQ. 1-NFLAG)) .AND. (0 .LT. 1+I-KKK) .AND. (0 .N . IFLAG)) .IMPLIES. ((0 .LE. -2+MLENGT))

13

```
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
      IVECT(I)=SPACE(K)
      (IVECT(I) .EQ. Q(45))
25    IFLAG=1
26    CONTINUE
```

```
      CALL DELETA(K,0)
      (K .EQ. 0)
3     WRITE(6,4)(IVECT(J),J=1,I)
4     FORMAT(1X,130A1)
COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
```

```
((0 .LE. -I+KKK) .AND. (0 .LT.
MLENGT-MSTART) .AND. (0 .LT. MSTART)
.AND. ( .NOT. ((0 .EQ. 4-I+MSTART)
.AND. (0 .EQ. 1-NFLAG))) .AND. (0
.EQ. Q(45)-SPACE(K)) .AND. (0 .EQ.
K))
 .IMPLIES.
((ALL(0 .EQ.
IVECT(JJ)-OUTPUT,JJ,1,I)))
```

```
14
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
       IVECT(I)=SPACE(K)
       (IVECT(I) .EQ. Q(45))
25     IFLAG=1
26     CONTINUE
       CALL DELETA(K,0)
        .NOT. (K .EQ. 0)
2      I=I+1
       (I .LE. KKK)
COND I .LT. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
```

NONE

```
15
COND I .LT. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
       IVECT(I)=SPACE(K)
       (IVECT(I) .EQ. Q(45))
25     IFLAG=1
26     CONTINUE
       CALL DELETA(K,0)
        .NOT. (K .EQ. 0)
2      I=I+1
        .NOT. (I .LE. KKK)
       I=KKK
       (IFLAG .EQ. 0)
3      WRITE(6,4)(IVECT(J),J=1,I)
4      FORMAT(1X,130A1)
COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
```

```
((0 .LE. -I+KKK) .AND. (0 .LT.
MLENGT-MSTART) .AND. (0 .LT. MSTART)
.AND. ( .NOT. ((0 .EQ. 4-I+MSTART)
.AND. (0 .EQ. 1-NFLAG))) .AND. (0
.EQ. Q(45)-SPACE(K)) .AND. (0 .NE.
K) .AND. (0 .LT. 1+I-KKK) .AND. (0
.EQ. 1)))
 .IMPLIES.
((ALL(0 .EQ.
```

```
                                                                    IVECT(JJ)-OUTPUT,JJ,1,KKK)))
16
      COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
      22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
             IVECT(I)=SPACE(K)
             (IVECT(I) .EQ. Q(45))
      25     IFLAG=1
      26     CONTINUE
             CALL DELETA(K,0)
              .NOT. (K .EQ. 0)
      2      I=I+1
              .NOT. (I .LE. KKK)
             I=KKK
              .NOT. (IFLAG .EQ. 0)
             IJ=I+1
             III=MLENGT-1
             II=1
      COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
                                                                    ((0 .LE. -I+KKK) .AND. (0 .LT.
                                                                    MLENGT-MSTART) .AND. (0 .LT. MSTART)
                                                                    .AND. ( .NOT. ((0 .EQ. 4-I+MSTART)
                                                                    .AND. (0 .EQ. 1-NFLAG))) .AND. (0
                                                                    .EQ. Q(45)-SPACE(K)) .AND. (0 .NE.
                                                                    K) .AND. (0 .LT. 1+I-KKK) .AND. (0
                                                                    .NE. 1))
                                                                     .IMPLIES.
                                                                    ((0 .LE. -2+MLENGT))
17
      COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
      22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
             IVECT(I)=SPACE(K)
              .NOT. (IVECT(I) .EQ. Q(45))
      26     CONTINUE
             CALL DELETA(K,0)
             (K .EQ. 0)
      3      WRITE(6,4)(IVECT(J),J=1,I)
      4      FORMAT(1X,130A1)
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
                                                                    ((0 .LE. -I+KKK) .AND. (0 .LT.
                                                                    MLENGT-MSTART) .AND. (0 .LT. MSTART)
                                                                    .AND. ( .NOT. ((0 .EQ. 4-I+MSTART)
                                                                    .AND. (0 .EQ. 1-NFLAG))) .AND. (0
                                                                    .NE. Q(45)-SPACE(K)) .AND. (0 .EQ.
                                                                    K))
                                                                     .IMPLIES.
                                                                    ((ALL(0 .EQ.
                                                                    IVECT(JJ)-OUTPUT,JJ,1,I)))
18
      COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
      22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
```

```
      IVECT(I)=SPACE(K)
       .NOT. (IVECT(I) .EQ. Q(45))
   26 CONTINUE
      CALL DELETA(K,0)
       .NOT. (K .EQ. 0)
    2 I=I+1
       (I .LE. KKK)
```

COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0

NONE

19

```
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
      IVECT(I)=SPACE(K)
       .NOT. (IVECT(I) .EQ. Q(45))
26    CONTINUE
      CALL DELETA(K,0)
       .NOT. (K .EQ. 0)
2     I=I+1
       .NOT. (I .LE. KKK)
      I=KKK
      (IFLAG .EQ. 0)
3     WRITE(6,4)(IVECT(J),J=1,I)
4     FORMAT(1X,130A1)
COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
```

((0 .LE. -I+KKK) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART) .AND. (.NOT. ((0 .EQ. 4-I+MSTART) .AND. (0 .EQ. 1-NFLAG))) .AND. (0 .NE. Q(45)-SPACE(K)) .AND. (0 .NE. K) .AND. (0 .LT. 1+I-KKK) .AND. (0 .EQ. IFLAG)) .IMPLIES. ((ALL(0 .EQ. IVECT(JJ)-OUTPUT,JJ,1,KKK)))

20

```
COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
22     .NOT. (I .EQ. MSTART+4 .AND. NFLAG .EQ. 1)
      IVECT(I)=SPACE(K)
       .NOT. (IVECT(I) .EQ. Q(45))
26    CONTINUE
      CALL DELETA(K,0)
       .NOT. (K .EQ. 0)
2     I=I+1
       .NOT. (I .LE. KKK)
      I=KKK
       .NOT. (IFLAG .EQ. 0)
      IJ=I+1
      III=MLENGT-1
      II=1
COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
```

((0 .LE. -I+KKK) .AND. (0 .LT. MLENGT-MSTART) .AND. (0 .LT. MSTART)

```
                                                                      .AND. ( .NOT. ((0 .EQ. 4-I+MSTART)
                                                                      .AND. (0 .EQ. 1-NFLAG))) .AND. (0
                                                                      .NE. Q(45)-SPACE(K)) .AND. (0 .NE.
                                                                      K) .AND. (0 .LT. 1+I-KKK) .AND. (0
                                                                      .NE. IFLAG))
                                                                       .IMPLIES.
                                                                      ((0 .LE. -2+MLENGT))

21
      COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
      6      (IVECT(IJ-II) .EQ. Q(45))
      3      WRITE(6,4)(IVECT(J),J=1,I)
      4      FORMAT(1X,130A1)
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
                                                                      ((0 .LE. -II+III) .AND. (0 .LE.
                                                                      -I+KKK) .AND. (0 .LT. MLENGT-MSTART)
                                                                      .AND. (0 .LT. MSTART) .AND. (0 .EQ.
                                                                      -IVECT(-II+IJ)+Q(45)))
                                                                       .IMPLIES.
                                                                      ((ALL(0 .EQ.
                                                                      IVECT(JJ)-OUTPUT,JJ,1,I)))

22
      COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
      6       .NOT. (IVECT(IJ-II) .EQ. Q(45))
              CALL INSERT(K,0,IVECT(IJ-II))
              I=I-1
              II=II+1
              (II .LE. III)
      COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
                                                                      NONE

23
      COND II .LE. III .AND. I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
      6       .NOT. (IVECT(IJ-II) .EQ. Q(45))
              CALL INSERT(K,0,IVECT(IJ-II))
              I=I-1
              II=II+1
               .NOT. (II .LE. III)
      3       WRITE(6,4)(IVECT(J),J=1,I)
      4       FORMAT(1X,130A1)
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
                                                                      ((0 .LE. -II+III) .AND. (0 .LE.
                                                                      -I+KKK) .AND. (0 .LT. MLENGT-MSTART)
                                                                      .AND. (0 .LT. MSTART) .AND. (0 .NE.
                                                                      -IVECT(-II+IJ)+Q(45)) .AND. (0 .LT.
                                                                      1+II-III))
                                                                       .IMPLIES.
                                                                      ((ALL(0 .EQ.
                                                                      IVECT(JJ)-OUTPUT,JJ,1,-1+I)))

24
```

```
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
     .     (NFLAG .EQ. 1)
     27     NSTART=MSTART+5
     28     CONTINUE
            (K .GT. 0)
     5      IFLAG=0
            I=NSTART
      COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
                                                                  ((0 .LE. -I+KKK) .AND. (ALL(0 .EQ.
                                                                  IVECT(JJ)-OUTPUT,JJ,1,I)) .AND. (0
                                                                  .EQ. 1-NFLAG) .AND. (0 .LT. K))
                                                                   .IMPLIES.
                                                                  ((0 .LT. MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LE.
                                                                  -5+KKK-MSTART))

     25
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
     .     (NFLAG .EQ. 1)
     27     NSTART=MSTART+5
     28     CONTINUE
             .NOT. (K .GT. 0)
      COND K .LE. 0 .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
                                                                  ((0 .LE. -I+KKK) .AND. (ALL(0 .EQ.
                                                                  IVECT(JJ)-OUTPUT,JJ,1,I)) .AND. (0
                                                                  .EQ. 1-NFLAG) .AND. (0 .LE. -K))
                                                                   .IMPLIES.
                                                                  ((ALL(0 .EQ.
                                                                  IVECT(JJ)-OUTPUT,JJ,1,I)))

     26
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
     .     .NOT. (NFLAG .EQ. 1)
     23     CONTINUE
            (K .GT. 0)
     5      IFLAG=0
            I=NSTART
      COND I .LE. KKK .AND. MLENGT .GT. MSTART .AND. MSTART .GT. 0
                                                                  ((0 .LE. -I+KKK) .AND. (ALL(0 .EQ.
                                                                  IVECT(JJ)-OUTPUT,JJ,1,I)) .AND. (0
                                                                  .NE. 1-NFLAG) .AND. (0 .LT. K))
                                                                   .IMPLIES.
                                                                  ((0 .LT. MSTART) .AND. (0 .LT.
                                                                  MLENGT-MSTART) .AND. (0 .LE.
                                                                  KKK-NSTART))

     27
      COND I .LE. KKK .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
     .     .NOT. (NFLAG .EQ. 1)
     28     CONTINUE
             .NOT. (K .GT. 0)
      COND K .LE. 0 .AND. ALL((OUTPUT .EQ. IVECT(JJ)),JJ,1,I)
                                                                  ((0 .LE. -I+KKK) .AND. (ALL(0 .EQ.
```

```
IVECT(JJ)-OUTPUT,JJ,1,I)) .AND. (0
.NE. 1-NFLAG) .AND. (0 .LE. -K))
 .IMPLIES.
((ALL(0 .EQ.
IVECT(JJ)-OUTPUT,JJ,1,I)))
```

2.18) JOUTPU(LINE,MSTART,MLENGT)

JOUTPU first creates a new string K which is just a copy of the string LINE. Then is calls SUBST, with K as the argument. The result of the call is that tokens for certain relational and logical operators are replaced with strings that can be more easily read. The resulting string K is printed by JOUTPU, beginning at the position MSTART of the line and going to the next line after MLENGT characters or after the first blank before MLENGT-th character if the MLENGT-th character is not a blank.

VERIFICATION:

Most of the paths are readily verifiable, except paths 11,13,15,17,19,21,23,25,27 since these paths involves output statements that are not implemented into the verifier.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
COND   LV(NAME,L1)
      SUBROUTINE LAST(NAME)
      COMMON/ALPHNU/Q(63)
      INTEGER Q
      COMMON/BLOCK/ISPACE(10000),IFAIL,IFREE
      ITEST = ISPACE(NAME+1)
      IF(ITEST) 20,10,20
 10   ISPACE(NAME+1) = IFREE
      IFREE = NAME
      NAME = 0
COND   LV(NAME,L1),L1=NIL
      RETURN
 20   JTEST = NAME
 30   ITEST = ISPACE(JTEST+1)
      IF(ISPACE(ITEST+1)) 40,50,40
COND   SPACE(ITEST) .NE. ELT(LENGTH(L1),L1)
 40   JTEST = ITEST
      GO TO 30
COND  ISPACE(ITEST)=ELT(LENGTH(L1),L1)
 50   ISPACE(ITEST+1) = IFREE
      IFREE  = ITEST
      ISPACE(JTEST+1) = 0
COND   SPACE(IFREE)=ELT(LENGTH(L1),L1)
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(NAME,L1) SUBROUTINE LAST(NAME) COMMON/ALPHNU/Q(63) INTEGER Q COMMON/BLOCK/ISPACE(10000),IFAIL,IFREE ITEST=ISPACE(NAME+1)	[ITEST]	[ISPACE,NAME]
IF(ITEST .NE. 0)GO TO 20 10 ISPACE(NAME+1)=IFREE IFREE=NAME NAME=0	[ISPACE] [IFREE] [NAME]	[ISPACE,NAME,IFREE] [NAME] []
COND LV(NAME,L1) .AND. L1=NIL RETURN		
20 JTEST=NAME	[JTEST]	[NAME]
30 ITEST=ISPACE(JTEST+1) IF(ISPACE(ITEST+1) .EQ. 0)GO TO 50	[ITEST]	[ISPACE,JTEST]
COND SPACE(ITEST) .NE. ELT(LENGTH(L1),L1) 40 JTEST=ITEST	[JTEST]	[ITEST]
GO TO 30		
COND ISPACE(ITEST)=ELT(LENGTH(L1),L1) 50 ISPACE(ITEST+1)=IFREE	[ISPACE] [IFREE]	[ISPACE,ITEST,IFREE] [ITEST]
IFREE=ITEST ISPACE(JTEST+1)=0	[ISPACE]	[ISPACE,JTEST]
COND SPACE(IFREE)=ELT(LENGTH(L1),L1) RETURN END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(NAME,L1) SUBROUTINE LAST(NAME) COMMON/ALPHNU/Q(63) INTEGER Q COMMON/BLOCK/ISPACE(10000),IFAIL,IFREE ITEST=ISPACE(NAME+1) (ITEST .NE. 0) 20 JTEST=NAME 30 ITEST=ISPACE(JTEST+1) (ISPACE(ITEST+1) .EQ. 0) COND ISPACE(ITEST)=ELT(LENGTH(L1),L1)	((LV(NAME,L1)) .AND. (0 .NE. -ISPACE(1+NAME)) .AND. (0 .EQ. ISPACE(1+ISPACE(1+NAME)))) .IMPLIES. ((0 .EQ. ELT(LENGTH(L1),L1)-ISPACE(ISPACE(1+NAME))))
2	COND LV(NAME,L1) SUBROUTINE LAST(NAME) COMMON/ALPHNU/Q(63) INTEGER Q COMMON/BLOCK/ISPACE(10000),IFAIL,IFREE ITEST=ISPACE(NAME+1) (ITEST .NE. 0) 20 JTEST=NAME 30 ITEST=ISPACE(JTEST+1) .NOT. (ISPACE(ITEST+1) .EQ. 0) COND SPACE(ITEST) .NE. ELT(LENGTH(L1),L1)	((LV(NAME,L1)) .AND. (0 .NE. -ISPACE(1+NAME)) .AND. (0 .NE. ISPACE(1+ISPACE(1+NAME)))) .IMPLIES. ((0 .NE. ELT(LENGTH(L1),L1)-ISPACE(ISPACE(1+NAME))))
3	COND LV(NAME,L1) SUBROUTINE LAST(NAME) COMMON/ALPHNU/Q(63) INTEGER Q COMMON/BLOCK/ISPACE(10000),IFAIL,IFREE ITEST=ISPACE(NAME+1) .NOT. (ITEST .NE. 0) 10 ISPACE(NAME+1)=IFREE IFREE=NAME NAME=0 COND LV(NAME,L1) .AND. L1=NIL	((LV(NAME,L1)) .AND. (0 .EQ. -ISPACE(1+NAME))) .IMPLIES. ((LV(0,L1)) .AND. (0 .EQ. -L1+NIL))

```
4
COND SPACE(ITEST) .NE. ELT(LENGTH(L1),L1)
40    JTEST=ITEST
30    ITEST=ISPACE(JTEST+1)
      (ISPACE(ITEST+1) .EQ. 0)
COND ISPACE(ITEST)=ELT(LENGTH(L1),L1)
                                   ((0 .NE. ELT(LENGTH(L1),L1)-SPACE(ITEST)) .AND. (0 .EQ.
                                   ISPACE(1+ISPACE(1+ITEST))))
                                    .IMPLIES.
                                   ((0 .EQ. ELT(LENGTH(L1),L1)-ISPACE(ISPACE(1+ITEST))))

5
COND SPACE(ITEST) .NE. ELT(LENGTH(L1),L1)
40    JTEST=ITEST
30    ITEST=ISPACE(JTEST+1)
       .NOT. (ISPACE(ITEST+1) .EQ. 0)
COND SPACE(ITEST) .NE. ELT(LENGTH(L1),L1)
                                   ((0 .NE. ELT(LENGTH(L1),L1)-SPACE(ITEST)) .AND. (0 .NE.
                                   ISPACE(1+ISPACE(1+ITEST))))
                                    .IMPLIES.
                                   ((0 .NE. ELT(LENGTH(L1),L1)-SPACE(ISPACE(1+ITEST))))

6
COND ISPACE(ITEST)=ELT(LENGTH(L1),L1)
50    ISPACE(ITEST+1)=IFREE
      IFREE=ITEST
      ISPACE(JTEST+1)=0
COND SPACE(IFREE)=ELT(LENGTH(L1),L1)
                                   ((0 .EQ. ELT(LENGTH(L1),L1)-ISPACE(ITEST)))
                                    .IMPLIES.
                                   ((0 .EQ. ELT(LENGTH(L1),L1)-SPACE(ITEST)))
```

2.19) LAST(NAME)

This subroutine merely deletes the last character of the string NAME and returns that cell to the string of available space.

VERIFICATION:

Path 1: Since IV(NAME,L1) and ISPACE(NAME+1)≠0, hence ISPACE(ISPACE(NAME+1))=ELT(I,L1). Since ISPACE(ISPACE(NAME+1)+1)=0, therefore ISPACE(ISPACE(NAME+1))=ELT(LENGTH(L1),L1).

Path 2: Similar to the above path, except that ISPACE(ISPACE(NAME+1)+1)≠0. Hence ISPACE(ISPACE(NAME+1))≠Elt(LENGTH(L1),L1).

Path 3: insufficient assertion.

Path 4: Similar to path 1 and 2.

Path 5: similar to path 4.

Path 6: obvious.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=(CDR(CDRN(I-1,L)).
COND   LV(SLOCL,L),NUM .GE. 0, LV(SLOCL,CDRN(-1,L))
      SUBROUTINE LEN (SLOCL,NUM,ENDL)
      COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
      INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL
      OLDP = SLOCL
      P = SLOCL
      N = 0
      IF(NUM) 6,4,6
    6 IF(SLOCL) 1,5,1
    1 N = N+1
      OLDP = P
      IF(SPACE(P+1)) 2,3,2
    2 P = SPACE(P+1)
COND   LV(OLDP,CDRN(N-1,L)),P=SPACE(OLDP+1),NUM .GE. N
      IF(NUM-N) 4,4,1
    3 IF(NUM-N) 5,4,5
    4 FAIL = 0
      ENDL = OLDP
COND   LV(ENDL,CDRN(NUM-1,L))
      RETURN
    5 FAIL = 1
      ENDL = OLDP
COND   FAIL=1
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(SLOCL,L) .AND. NUM .GE. 0 .AND. LV(SLOCL,CORN(-1,L))		
SUBROUTINE LEN (SLOCL,NUM,ENDL)		
COMMON/BLOCK/SPACE(10000),FAIL,IFQQ		
INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL		
OLDP=SLOCL	[OLDP]	[SLOCL]
P=SLOCL	[P]	[SLOCL]
N=0	[N]	[]
IF(NUM .EQ. 0)GO TO 4		
6 IF(SLOCL .EQ. 0)GO TO 5		
1 N=N+1	[N]	[N]
OLDP=P	[OLDP]	[P]
IF(SPACE(P+1) .EQ. 0)GO TO 3		
2 P=SPACE(P+1)	[P]	[SPACE,P]
COND LV(OLDP,CORN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N		
IF(NUM-N .LE. 0)GO TO 4		
GO TO 1		
3 IF(NUM-N .NE. 0)GO TO 5		
4 FAIL=0	[FAIL]	[]
ENDL=OLDP	[ENDL]	[OLDP]
COND LV(ENDL,CORN(NUM-1,L))		
RETURN		
5 FAIL=1	[FAIL]	[]
ENDL=OLDP	[ENDL]	[OLDP]
COND FAIL=1		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(SLOCL,L) .AND. NUM .GE. 0 .AND. LV(SLOCL,CDRN(-1,L)) SUBROUTINE LEN (SLOCL,NUM,ENDL) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL OLDP=SLOCL P=SLOCL N=0 (NUM .EQ. 0) 4 FAIL=0 ENDL=OLDP COND LV(ENDL,CDRN(NUM-1,L))	((LV(SLOCL,L)) .AND. (0 .LE. NUM) .AND. (LV(SLOCL,CDRN(-1,L))) .AND. (0 .EQ. NUM)) .IMPLIES. ((LV(SLOCL,CDRN(-1+NUM,L))))
2	COND LV(SLOCL,L) .AND. NUM .GE. 0 .AND. LV(SLOCL,CDRN(-1,L)) SUBROUTINE LEN (SLOCL,NUM,ENDL) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL OLDP=SLOCL P=SLOCL N=0 .NOT. (NUM .EQ. 0) 6 (SLOCL .EQ. 0) 5 FAIL=1 ENDL=OLDP COND FAIL=1	NONE
3	COND LV(SLOCL,L) .AND. NUM .GE. 0 .AND. LV(SLOCL,CDRN(-1,L)) SUBROUTINE LEN (SLOCL,NUM,ENDL) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL OLDP=SLOCL P=SLOCL N=0 .NOT. (NUM .EQ. 0) 6 .NOT. (SLOCL .EQ. 0) 1 N=N+1 OLDP=P (SPACE(P+1) .EQ. 0) 3 (NUM-N .NE. 0) 5 FAIL=1 ENDL=OLDP COND FAIL=1	

NONE

```
4
COND LV(SLOCL,L) .AND. NUM .GE. 0 .AND. LV(SLOCL,CDRN(-1,L))
      SUBROUTINE LEN (SLOCL,NUM,ENDL)
      COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
      INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL
      OLDP=SLOCL
      P=SLOCL
      N=0
        .NOT. (NUM .EQ. 0)
6       .NOT. (SLOCL .EQ. 0)
1     P=N+1
      OLDP=P
      (SPACE(P+1) .EQ. 0)
3       .NOT. (NUM-N .NE. 0)
4     FAIL=0
      ENDL=OLDP
COND LV(ENDL,CDRN(NUM-1,L))
                                        ((LV(SLOCL,L)) .AND. (0 .LE. NUM) .AND.
                                        (LV(SLOCL,CDRN(-1,L))) .AND. (0 .NE. NUM) .AND. (0
                                        .NE. SLOCL) .AND. (0 .EQ. SPACE(1+SLOCL)) .AND. (0
                                        .EQ. 1-NUM))
                                         .IMPLIES.
                                        ((LV(SLOCL,CDRN(-1+NUM,L))))

5
COND LV(SLOCL,L) .AND. NUM .GE. 0 .AND. LV(SLOCL,CDRN(-1,L))
      SUBROUTINE LEN (SLOCL,NUM,ENDL)
      COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
      INTEGER SLOCL,NUM,ENDL,OLDP,P,N,SPACE,FAIL
      OLDP=SLOCL
      P=SLOCL
      N=0
        .NOT. (NUM .EQ. 0)
6       .NOT. (SLOCL .EQ. 0)
1     P=N+1
      OLDP=P
        .NOT. (SPACE(P+1) .EQ. 0)
2     P=SPACE(P+1)
COND LV(OLDP,CDRN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N
                                        ((LV(SLOCL,L)) .AND. (0 .LE. NUM) .AND.
                                        (LV(SLOCL,CDRN(-1,L))) .AND. (0 .NE. NUM) .AND. (0
                                        .NE. SLOCL) .AND. (0 .NE. SPACE(1+SLOCL)))
                                         .IMPLIES.
                                        ((LV(SLOCL,CDRN(0,L))) .AND. (0 .LE. -1+NUM))

6
COND LV(OLDP,CDRN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N
      (NUM-N .LE. 0)
4     FAIL=0
      ENDL=OLDP
COND LV(ENDL,CDRN(NUM-1,L))
```

```
                                                    ((LV(OLDP,CDRN(-1+N,L))) .AND. (0 .EQ.
                                                    -P+SPACE(1+OLDP)) .AND. (0 .LE. -N+NUM) .AND. (0 .LE.
                                                    N-NUM))
                                                     .IMPLIES.
                                                    ((LV(OLDP,CDRN(-1+NUM,L))))

7
      COND LV(OLDP,CDRN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N
             .NOT. (NUM-N .LE. 0)
      1      N=N+1
             OLDP=P
             (SPACE(P+1) .EQ. 0)
      3      (NUM-N .NE. 0)
      5      FAIL=1
             ENDL=OLDP
      COND FAIL=1
                                                    NONE

8
      COND LV(OLDP,CDRN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N
             .NOT. (NUM-N .LE. 0)
      1      N=N+1
             OLDP=P
             (SPACE(P+1) .EQ. 0)
      3       .NOT. (NUM-N .NE. 0)
      4      FAIL=0
             ENDL=OLDP
      COND LV(ENDL,CDRN(NUM-1,L))
                                                    ((LV(OLDP,CDRN(-1+N,L))) .AND. (0 .EQ.
                                                    -P+SPACE(1+OLDP)) .AND. (0 .LE. -N+NUM) .AND. (0 .LT.
                                                    -N+NUM) .AND. (0 .EQ. SPACE(1+P)) .AND. (0 .EQ.
                                                    1+N-NUM))
                                                     .IMPLIES.
                                                    ((LV(P,CDRN(-1+NUM,L))))

9
      COND LV(OLDP,CDRN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N
             .NOT. (NUM-N .LE. 0)
      1      N=N+1
             OLDP=P
              .NOT. (SPACE(P+1) .EQ. 0)
      2      P=SPACE(P+1)
      COND LV(OLDP,CDRN(N-1,L)) .AND. P=SPACE(OLDP+1) .AND. NUM .GE. N
                                                    ((LV(OLDP,CDRN(-1+N,L))) .AND. (0 .EQ.
                                                    -P+SPACE(1+OLDP)) .AND. (0 .LE. -N+NUM) .AND. (0 .LT.
                                                    -N+NUM) .AND. (0 .NE. SPACE(1+P)))
                                                     .IMPLIES.
                                                    ((LV(P,CDRN(N,L))))
```

2.20) LEN(SLOCL,NUM,ENDL)

This subroutine works like the LEN function of SNOBOL. It sets ENDL to the NUM-th character of the string SLOCL.

example: SLOCL = "27TH" and NUM = 3
result : ENDL is a pointer to the third character "T" in the string "27TH".

VERIFICATION:

PATH 1: Since NUM=0, the conclusion is true.

PATH 2: proved.

Path 3: proved.

PATH 4: Since NUM=1, hence LV(SLOCL,CDRN(-1+NUM,L)) becomes LV(SLOCL,L) which is true.

PATH 5: (0 .LE. NUM) and (.NOT.(0=NUM)) implies (0 .LE. -1+NUM). Since CDRN(0,L)=L, hence LV(SLOCL,CDRN(0,L)) is true.

PATH 6: (0 .LE. -N+NUM) and (0 .LE. N-NUM) implies N=NUM. Therefore LV(OLDP,CDRN(-1+NUM,L)) becomes LV(OLDP,CDRN(-1+N,L)) which implied by the initial assertion.

PATH 7: proved.

PATH 8: Since P=SPACE(1+OLDP) and LV(OLDP,CDRN(-1+N,L)), hence LV(P,CDRN(N,L)). Since NUM=N+1, therefore LV(P,CDRN(-1+NUM,L)).

PATH 9: (0 .LE. N-NUM) is false, hence (0 .GT. N-NUM) which proves (0 .LE. -1-N+NUM). LV(P,CDRN(N,L)) is proved in the same way as in path 8.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), COR(L)).
COND   LV(FREE,L)
       INTEGER FUNCTION NEWCEL(IDUMMY)
CC SPACE(N)
       COMMON/BLOCK/SPACE(10000),F,FREE
CC SPACE(N)
       INTEGER SPACE,FREE,F
       IF (FREE - 0) 100,100,200
CCC
100    PRINT 1
1      FORMAT(1H1,20X,27H*****OVERFLOW OF SPACE*****)
COND    FREE .LE. 0, L=NIL
       STOP
200    NEWCEL = FREE
       FREE = SPACE(FREE + 1)
COND    SPACE(NEWCEL)=CAR(L), LV(FREE,COR(L))
       RETURN
       END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(FREE,L) INTEGER FUNCTION NEWCEL(IDUMMY) COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,FREE,F IF(FREE-0 .GT. 0)GO TO 200 100 PRINT 1 1 FORMAT(1H1,20X,27HHHH-OVERFLOW OF SPACEUU*) COND FREE .LE. 0 .AND. L=NIL STOP 200 NEWCEL=FREE	[NEWCEL]	[FREE]
FREE=SPACE(FREE+1) COND SPACE(NEWCEL)=CAR(L) .AND. LV(FREE,CDR(L)) RETURN END	[FREE]	[SPACE,FREE]

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(FREE,L) INTEGER FUNCTION NEWCEL(IDUMMY) COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,FREE,F (FREE=0 .GT. 0) 200 NEWCEL=FREE FREE=SPACE(FREE+1) COND SPACE(NEWCEL)=CAR(L) .AND. LV(FREE,CDR(L))	((LV(FREE,L)) .AND. (0 .LT. FREE)) .IMPLIES. ((LV(SPACE(1+FREE),CDR(L))) .AND. (0 .EQ. CAR(L)-SPACE(FREE)))
2	COND LV(FREE,L) INTEGER FUNCTION NEWCEL(IDUMMY) COMMON/BLOCK/SPACE(10000),F,FREE INTEGER SPACE,FREE,F .NOT. (FREE=0 .GT. 0) 100 PRINT 1 1 FORMAT(1H1,20X,27HUU*OVERFLOW OF SPACEUU*) COND FREE .LE. 0 .AND. L=NIL	((LV(FREE,L)) .AND. (0 .LE. -FREE)) .IMPLIES. ((0 .EQ. -L+NIL))

2.21) NEWCEL(JUNKEY)

NEWCEL removes a cell from the string of available space and returns a pointer to that cell. It also checks for overflows, when the string of available space is empty.

VERIFICATION:

PATH 1: Since LV(FREE,L), therefore SPACE(FREE)=CAR(L) and LV(SPACE(1+FREE),CDR(L)) holds.

PATH 2: By the properties of LIST this path is true.

```
CC     LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC     SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CCND    LV(FREE,L)
       SUBROUTINE NLIST(STRING,DATA)
CC SPACE(N)
       COMMON/BLOCK/SPACE(10000),I00,I000
CC SPACE(N)
       INTEGER SPACE,STRING,DATA
C ACQUIRE A NEW CELL FROM FREE
       STRING = NEWCEL(0)
CCND    CAR(L)=SPACE(STRING),LV(FREE,CDR(L))
       SPACE(STRING+1) = 0
       SPACE(STRING) = DATA
CCND    LV(STRING,L1),SPACE(STRING)=DATA
       RETURN
       END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(FREE,L) SUBROUTINE NLIST(STRING,DATA) COMMON/BLOCK/SPACE(10000),IQQ,IQQQ INTEGER SPACE,STRING,DATA STRING=NEWCEL(0)	[STRING]	[NEWCEL]
COND CDR(L)=SPACE(STRING) .AND. LV(FREE,CDR(L)) SPACE(STRING+1)=0 SPACE(STRING)=DATA	[SPACE] [SPACE]	[SPACE,STRING] [SPACE,STRING,DATA]
COND LV(STRING,L1) .AND. SPACE(STRING)=DATA RETURN END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND.LV(FREE,L) SUBROUTINE NLIST(STRING,DATA) COMMON/BLOCK/SPACE(10000),IQQ,IQQQ INTEGER SPACE,STRING,DATA STRING=NEWCEL(0) COND CAR(L)=SPACE(STRING) .AND. LV(FREE,CDR(L))	((LV(FREE,L))) .IMPLIES. ((0 .EQ. -CAR(L)+SPACE(NEWCEL(0))) .AND. (LV(FREE,CDR(L))))
2	COND CAR(L)=SPACE(STRING) .AND. LV(FREE,CDR(L)) SPACE(STRING+1)=0 SPACE(STRING)=DATA COND LV(STRING,L1) .AND. SPACE(STRING)=DATA	((0 .EQ. -CAR(L)+SPACE(STRING)) .AND. (LV(FREE,CDR(L)))) .IMPLIES. ((LV(STRING,L1)))
3		
2		
1		

2.22) NLIST(STRING,DATA)

This subroutine is called in order to create a new string STRING with the character DATA in it.

example: DATA = 1F*
result : STRING = "*"

VERIFICATION:

PATH 1: Assertion is insufficient. The function NEWCVI has to be defined.

PATH 2: Since SPACE(STRING)=CAR(L), therefore STRING is the initial pointer of L. Hence LV(STRING,L) is true.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC    .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC    LIST.PIR(P,L)=EXISTS(P=ADDR(K,L),K,1,LENGTH(L)).
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    ALL(F,X,I,J)=F(X), X .IN. (I..J).
CC    EXISTS(F,X,I,J)=F(X), I .LE. X, X.LE. J.
CC    ALL(EXISTS(J,A,X,I,B,Y))=ALL(EXISTS(ELT(I,L1)=ELT(J,L2),
CC    J,A,LENGTH(X)),I,B,LENGTH(Y))
CC    SL=SUBLIST
CCND   LV(SLOCNA,L1),LV(PLOCNA,L2),(ANC=0 .OR. ANC=1)
      SUBROUTINE MOTARY(SLOCNA,PLOCNA,BEGNA,ANC)
      INTEGER SLOCNA,PLOCNA,BEGNA,ANC,FAIL,OLDSP,SP,PP,SPACE
      COMMON/BLOCK/SPACE(10000),FAIL,IFGO
      OLDSP = SLOCNA
      SP = SLOCNA
      BEGNA = SLOCNA
      IF(SLOCNA) 1,7,1
    1 PP = PLOCNA
    2 IF(SPACE(SP)-SPACE(PP)) 4,3,4
COND   ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SPACE(SP+1),L1))),
COND   (ANC=0 .OR. ANC=1), LIST.PIR(PP,L2)
    3 IF(ANC) 5,5,7
    5 OLDSP = SP
      SP = SPACE(SP+1)
      IF(SP) 1,7,1
COND   ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))),(ANC=0 .OR. ANC=1),
COND   ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,
COND   SPACE(PP+1),L2))),LIST.PIR(PP,L2)
    4 PP = SPACE(PP+1)
      IF(PP) 2,5,2
    6 FAIL = 1
      BEGNA = SP
COND   ALL(EXISTS(J,1,L2,SL(SLOCAN,BEGNA,L1))),
COND   ALL((SPACE(BEGNA) .NE. ELT(J,L2)),J,1,LENGTH(L2)),FAIL=0
      RETURN
    7 FAIL = 1
      BEGNA = OLDSP
COND   (ALL(EXISTS(I,1,L2,I,1,L1)),(FAIL=1))
COND   .OR. ((ANC=1),(FAIL=1),(EXISTS((CAR(L1)=ELT(J,L2)),J,1,
COND   LENGTH(L2))))
      RETURN
      END
```

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(SLOCNA,L1) .AND. LV(PLOCNA,L2) .AND. (ANC .EQ. 0 .OR. ANC .EQ. 1) SUBROUTINE NOTANY(SLOCNA,PLOCNA,BEGNA,ANC) INTEGER SLOCNA,PLOCNA,BEGNA,ANC,FAIL,OLDSP,SP,PP,SPACE COMMON/BLOCK/SPACE(10000),FAIL,IFQQ OLDSP=SLOCNA SP=SLOCNA BEGNA=SLOCNA (SLOCNA .EQ. 0) 7 FAIL=1 BEGNA=OLDSP COND (ALL(EXISTS(J,1,L2,I,1,L1)),(FAIL .EQ. 1)) .OR. ((ANC .EQ. 1),(FAIL .EQ. COND 1),(EXISTS((CAR(L1) .EQ. ELT(J,L2)),J,1,LENGTH(L2))))	(((LV(SLOCNA,L1)) .AND. (LV(PLOCNA,L2)) .AND. ((0 .EQ. ANC) .OR. (0 .EQ. 1-ANC)) .AND. (0 .EQ. SLOCNA)) .IMPLIES. ((ALL(EXISTS(J,1,L2,I,1,L1)), .TRUE.) .OR. (0 .EQ. 1-ANC, .TRUE. ,EXISTS(0 .EQ. -CAR(L1)+ELT(J,L2),J,1,LENGTH(L2))))
2	COND LV(SLOCNA,L1) .AND. LV(PLOCNA,L2) .AND. (ANC .EQ. 0 .OR. ANC .EQ. 1) SUBROUTINE NOTANY(SLOCNA,PLOCNA,BEGNA,ANC) INTEGER SLOCNA,PLOCNA,BEGNA,ANC,FAIL,OLDSP,SP,PP,SPACE COMMON/BLOCK/SPACE(10000),FAIL,IFQQ OLDSP=SLOCNA SP=SLOCNA BEGNA=SLOCNA .OR. (SLOCNA .EQ. 0) 1 PP=PLOCNA 2 (SPACE(SP)-SPACE(PP) .NE. 0) COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR. ANC .EQ. COND 1) .AND. ALL((SPACE(SP) .NE. COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND. COND LIST.PIR(PP,L2)	(((LV(SLOCNA,L1)) .AND. (LV(PLOCNA,L2)) .AND. ((0 .EQ. ANC) .OR. (0 .EQ. 1-ANC)) .AND. (0 .NE. SLOCNA) .AND. (0 .NE. SPACE(PLOCNA)-SPACE(SLOCNA))) .IMPLIES. ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SLO CNA,L1)))) .AND. (LIST.PIR(PLOCNA,L2)) .AND. (ALL(0 .NE. ELT(J,L2)-SPACE(SLOCNA),J,1,LENGTH(SU BLIST(PLOCNA,SPACE(1+PLOCNA),L2))))

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(SLOCNA,L1) .AND. LV(PLOCNA,L2) .AND. (ANC .EQ. 0 .OR. ANC .EQ.		
COND 1)		
SUBROUTINE OBTAIN(SLOCNA,PLOCNA,BEGNA,ANC)		
INTEGER SLOCNA,PLOCNA,BEGNA,ANC,FAIL,OLDSP,SP,PP,SPACE		
COMMON/BLOCK/SPACE(10000),FAIL,IEQ		
OLDSP=SLOCNA	[OLDSP]	[SLOCNA]
SP=SLOCNA	[SP]	[SLOCNA]
BEGNA=SLOCNA	[BEGNA]	[SLOCNA]
IF(SLOCNA .EQ. 0)GO TO 7		
1 PP=PLOCNA	[PP]	[PLOCNA]
2 IF(SPACE(SP)-SPACE(PP) .NE. 0)GO TO 4		
COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ.		
COND 0 .OR. ANC .EQ. 1) .AND. LIST.PIR(PP,L2)		
3 IF(ANC .GT. 0)GO TO 7	[OLDSP]	[SP]
5 OLDSP=SP	[SP]	[SPACE,SP]
SP=SPACE(SP+1)		
IF(SP .NE. 0)GO TO 1		
GO TO 7		
COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR.		
COND ANC .EQ. 1) .AND. ALL((SPACE(SP) .NE.		
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND.		
COND LIST.PIR(PP,L2)		
4 PP=SPACE(PP+1)	[PP]	[SPACE,PP]
IF(PP .NE. 0)GO TO 2		
6 FAIL=0	[FAIL]	[]
BEGNA=SP	[BEGNA]	[SP]
COND ALL(EXISTS(J,1,L2,SL(SLOCAN,BEGNA,L1))) .AND. ALL((SPACE(BEGNA)		
COND .NE. ELT(J,L2)),J,1,LENGTH(L2)) .AND. FAIL=0		
RETURN		
7 FAIL=1	[FAIL]	[]
BEGNA=OLDSP	[BEGNA]	[OLDSP]
COND (ALL(EXISTS(J,1,L2,1,1,L1)),(FAIL .EQ. 1)) .OR. ((ANC .EQ.		
COND 1),(FAIL .EQ. 1),(EXISTS((CAR(L1) .EQ. ELT(J,L2)),J,1,LENGTH(L2))))		
RETURN		
END		

```
                                                                      .AND. ((0 .EQ. ANC) .OR. (0 .EQ.
                                                                      1-ANC)))

3
      COND LV(SLOCNA,L1) .AND. LV(PLOCNA,L2) .AND. (ANC .EQ. 0 .OR. ANC .EQ. 1)
           SUBROUTINE NOTANY(SLOCNA,PLOCNA,BEGNA,ANC)
           INTEGER SLOCNA,PLOCNA,BEGNA,ANC,FAIL,OLDSP,SP,PP,SPACE
           COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
           OLDSP=SLOCNA
           SP=SLOCNA
           BEGNA=SLOCNA
            .NOT. (SLOCNA .EQ. 0)
      1    PP=PLOCNA
      2     .NOT. (SPACE(SP)-SPACE(PP) .NE. 0)
      COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
      COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
                                                                      ((LV(SLOCNA,L1)) .AND.
                                                                      (LV(PLOCNA,L2)) .AND. ((0 .EQ. ANC)
                                                                      .OR. (0 .EQ. 1-ANC)) .AND. (0 .NE.
                                                                      SLOCNA) .AND. (0 .EQ.
                                                                      SPACE(PLOCNA)-SPACE(SLOCNA)))
                                                                       .IMPLIES.
                                                                      ((ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPA
                                                                      CE(1+SLOCNA),L1)))) .AND.
                                                                      (LIST.PIR(PLOCNA,L2)) .AND. ((0 .EQ.
                                                                      ANC) .OR. (0 .EQ. 1-ANC)))

4
      COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
      COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
      3     (ANC .GT. 0)
      7     FAIL=1
            BEGNA=OLDSP
      COND (ALL(EXISTS(J,1,L2,1,1,L1)),(FAIL .EQ. 1)) .OR. ((ANC .EQ. 1),(FAIL .EQ.
      COND 1),(EXISTS(I,(CAR(L1) .EQ. ELT(J,L2)),J,1,LENGTH(L2))))
                                                                      ((ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPA
                                                                      CE(1+SP),L1)))) .AND. ((0 .EQ. ANC)
                                                                      .OR. (0 .EQ. 1-ANC)) .AND.
                                                                      (LIST.PIR(PP,L2)) .AND. (0 .LT. ANC))
                                                                       .IMPLIES.
                                                                      ((ALL(EXISTS(J,1,L2,1,1,L1)), .TRUE.
                                                                      ) .OR. (0 .EQ. 1-ANC, .TRUE.
                                                                      ,EXISTS(J .EQ.
                                                                      -CAR(L1)+ELT(J,L2),J,1,LENGTH(L2))))

5
      COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
      COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
      3      .NOT. (ANC .GT. 0)
      5     OLDSP=SP
            SP=SPACE(SP+1)
            (SP .NE. 0)
      1     PP=PLOCNA
```

```
2       (SPACE(SP)-SPACE(PP) .NE. 0)
COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR. ANC .EQ.
COND 1) .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND.
COND LIST.PIR(PP,L2)
                                                                    ((ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPA
                                                                    CE(1+SP),L1)))) .AND. ((0 .EQ. ANC)
                                                                    .OR. (0 .EQ. 1-ANC)) .AND.
                                                                    (LIST.PIR(PP,L2)) .AND. (0 .LE.
                                                                    -ANC) .AND. (0 .NE. -SPACE(1+SP))
                                                                    .AND. (0 .NE.
                                                                    SPACE(PLOCNA)-SPACE(SPACE(1+SP))))
                                                                     .IMPLIES.
                                                                    ((ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPA
                                                                    CE(1+SP),L1)))) .AND.
                                                                    (LIST.PIR(PLOCNA,L2)) .AND. (ALL(0
                                                                    .NE.
                                                                    ELT(J,L2)-SPACE(SPACE(1+SP)),J,1,LENG
                                                                    TH(SUBLIST(PLOCNA,SPACE(1+PLOCNA),L2)
                                                                    ))) .AND. ((0 .EQ. ANC) .OR. (0 .EQ.
                                                                    1-ANC)))

6
COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
3       .NOT. (ANC .GT. 0)
5       OLDSP=SP
        SP=SPACE(SP+1)
        (SP .NE. 0)
1       PP=PLOCNA
2       .NOT. (SPACE(SP)-SPACE(PP) .NE. 0)
COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
                                                                    ((ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPA
                                                                    CE(1+SP),L1)))) .AND. ((0 .EQ. ANC)
                                                                    .OR. (0 .EQ. 1-ANC)) .AND.
                                                                    (LIST.PIR(PP,L2)) .AND. (0 .LE.
                                                                    -ANC) .AND. (0 .NE. -SPACE(1+SP))
                                                                    .AND. (0 .EQ.
                                                                    SPACE(PLOCNA)-SPACE(SPACE(1+SP))))
                                                                     .IMPLIES.
                                                                    ((ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPA
                                                                    CE(1+SPACE(1+SP)),L1)))) .AND.
                                                                    (LIST.PIR(PLOCNA,L2)) .AND. ((0 .EQ.
                                                                    ANC) .OR. (0 .EQ. 1-ANC)))

7
COND ALL(EXISTS(J,1,L2,1,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
3       .NOT. (ANC .GT. 0)
5       OLDSP=SP
        SP=SPACE(SP+1)
        .NOT. (SP .NE. 0)
7       FAIL=1
```

```
      BEGIN=FLOSP
COND (ALL(EXISTS(J,1,L2,I,1,L1)),(FAIL .EQ. 1)) .OR. ((ANC .EQ. 1),(FAIL .EQ.
COND 1),(EXISTS((CAR(L1) .EQ. ELT(J,L2)),J,1,LENGTH(L2))))
                                                    ((ALL(EXISTS(J,I,L2,I,1,SL(SLOCAN,SPA
                                                    CE(1+SP),L1)))) .AND. ((0 .EQ. ANC)
                                                    .OR. (0 .EQ. 1-ANC)) .AND.
                                                    (LIST.PIR(PP,L2)) .AND. (0 .LE.
                                                    -ANC) .AND. (0 .EQ. -SPACE(1+SP)))
                                                     .IMPLIES.
                                                    ((ALL(EXISTS(J,1,L2,I,1,L1)), .TRUE.
                                                    ) .OR. (0 .EQ. 1-ANC, .TRUE.
                                                    ,EXISTS(0 .EQ.
                                                    -CAR(L1)+ELT(J,L2),J,1,LENGTH(L2))))

8
COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR. ANC .EQ.
COND 1) .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND.
COND LIST.PIR(PP,L2)
4      PP=SPACE(PP+1)
       (PP .NE. 0)
2      (SPACE(SP)-SPACE(PP) .NE. 0)
COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR. ANC .EQ.
COND 1) .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND.
COND LIST.PIR(PP,L2)
                                                    ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,
                                                    L1)))) .AND. ((0 .EQ. ANC) .OR. (0
                                                    .EQ. 1-ANC)) .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCNA,SPACE(1+PP),L2)))) .AND.
                                                    (LIST.PIR(PP,L2)) .AND. (0 .NE.
                                                    -SPACE(1+PP)) .AND. (0 .NE.
                                                    SPACE(SPACE(1+PP))-SPACE(SP)))
                                                     .IMPLIES.
                                                    ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,
                                                    L1)))) .AND.
                                                    (LIST.PIR(SPACE(1+PP),L2)) .AND.
                                                    (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCNA,SPACE(1+SPACE(1+PP)),L2))))
                                                    .AND. ((0 .EQ. ANC) .OR. (0 .EQ.
                                                    1-ANC)))

9
COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR. ANC .EQ.
COND 1) .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND.
COND LIST.PIR(PP,L2)
4      PP=SPACE(PP+1)
       (PP .NE. 0)
2       .NOT. (SPACE(SP)-SPACE(PP) .NE. 0)
COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SPACE(SP+1),L1))) .AND. (ANC .EQ. 0 .OR.
COND ANC .EQ. 1) .AND. LIST.PIR(PP,L2)
```

```
                                                                ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,
                                                                L1)))) .AND. ((J .EQ. ANC) .OR. (0
                                                                .EQ. 1-ANC)) .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                                T(PLOCNA,SPACE(1+PP),L2)))) .AND.
                                                                (LIST.PTR(PP,L2)) .AND. (0 .NE.
                                                                -SPACE(1+PP)) .AND. (0 .EQ.
                                                                SPACE(SPACE(1+PP))-SPACE(SP)))
                                                                 .IMPLIES.
                                                                ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SPA
                                                                CE(1+SP),L1)))) .AND.
                                                                (LIST.PTR(SPACE(1+PP),L2)) .AND. ((0
                                                                .EQ. ANC) .OR. (0 .EQ. 1-ANC)))

10
      COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))) .AND. (ANC .EQ. 0 .OR. ANC .EQ.
      COND 1) .AND. ALL((SPACE(SP) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) .AND.
      COND LIST.PTR(PP,L2)
      4     PP=SPACE(PP+1)
             .NOT. (PP .NE. 0)
      6     FAIL=0
            BEGNA=SP
      COND ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,BEGNA,L1))) .AND. ALL((SPACE(BEGNA) .NE.
      COND ELT(J,L2)),J,1,LENGTH(L2)) .AND. FAIL=0
                                                                ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,
                                                                L1)))) .AND. ((0 .EQ. ANC) .OR. (0
                                                                .EQ. 1-ANC)) .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                                T(PLOCNA,SPACE(1+PP),L2)))) .AND.
                                                                (LIST.PTR(PP,L2)) .AND. (0 .EQ.
                                                                -SPACE(1+PP)))
                                                                 .IMPLIES.
                                                                ((ALL(EXISTS(J,1,L2,I,1,SL(SLOCAN,SP,L1))
                                                                )) .AND. (ALL(0 .NE.
                                                                ELT(J,L2)-SPACE(SP),J,1,LENGTH(L2))))
```

2.23) NOTANY(SLOCNA,PLOCNA,REGNA,ANC)

For ANC=0 REGNA is set to point to the first position in the string SLOCNA where none of the characters of the string PLOCNA matched. For ANC=1, only the first character of SLOCNA is checked.

example: SLOCNA="CONSONANTS", ANC=0, PLOCNA="AEIOU"
result : REGNA POINTS TO "C", the first character of SLOCNA.

VERIFICATION:

Path 1: Since SLOCNA=0, L1 is nil and therefore by definition the conclusion is true.

Path 2: Since SL(SLOCNA,SLOCNA,L1), SL1 is nil. SPACE(SLOCNA)≠SPACE(PLOCNA) and IV(PLOCNA,L2) implies ALL(SPACE(SLOCNA)≠ELT(J,L2),J,1,LENGTH (SUBLIST(PLOCNA,SPACE(PLOCNA+1),L2))).

Path 3: Since SPACE(PLOCNA)=SPACE(SLOCNA), hence ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SLOCNA+1), L1))).

Path 4: ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SP+1), L1))) implies EXISTS(CAR(L1)=ELT(J,L2),J,1,LENGTH(L2))

Path 5: ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SP+1),L1))) has not changed. SPACE(PLOCNA)=SPACE(SPACE(SP+1)) implies ALL(ELT(J,L2)=SPACE(SPACE(SP+1)),J,1, LENGTH(SUBLIST(PLOCNA,SPACE(PLOCNA+1),L2))).

Path 6: Since ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SP+1), L1))) and SPACE(SPACE(SP+1))=SPACE(PLOCNA), hence ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SPACE (SP+1)+1),L1))).

Path 7: Since ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SP+1), L1))) and SPACE(SP+1)=0, therefore ALL(EXISTS(J,1,L2,I,1,L1)) and EXISTS(CAR(L1)= ELT(J,L2),J,1,LENGTH(L2)).

Path 8: ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SP,L1))) is not changed.
Since ALL(ELT(J,L2)≠SPACE(SP),J,1,LENGTH(SUBLIST(PLOCNA,SPACE(PP+1),L2))) and SPACE(SP)≠SPACE(SPACE (PP+1)), hence ALL(ELT(J,L2)≠SPACE(SP),J,1,LENGTH (SUBLIST(PLOCNA,SPACE(SPACE(PP+1)+1),L2))).

Path 9: Since ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SP,L1)))
and SPACE(SP)=SPACE(SPACE(PP+1)), hence
ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SPACE(SP+1),
L1))).

Path 10: ALL(EXISTS(J,1,L2,I,1,SL(SLOCNA,SP,L1))) is
not changed.
ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(SUBLIST
(PLOCNA,SPACE(PP+1),L2))) and
SPACE(PP+1)=0
implies
ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(L2)).

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    EXISTS(F,X,I,J)=F(X), I .LE. X, X .LE. J.
CC    LIST.PIR(P,L)=EXISTS(P=ADDR(K,L),K,1,LENGTH(L)).
COND   EXISTS((SPACE(INDEX)=ELT(II+1,L)),II+1,1,LENGTH(L)),LV(STRING,L)
      INTEGER FUNCTION PREV(STRING,INDEX)
      INTEGER SPACE,STRING
      COMMON/BLOCK/SPACE(10000),I90,I999
C  THIS FUNCTION FINDS THE PREVIOUS CELL OF A STRING. FOR EXAMPLE, IF
C  AA IS A STRING AND IND1 AN ADDRESS SOMEWHERE IN STRING AA,
C        IND2=PREV(AA,IND1)  WILL PUT IN IND2 THE ADDRESS OF THE CELL
C  WHICH IS PREVIOUS TO IND1.  CAUTION: UPON RETURNING, IND2 SHOULD BE
C  CHECKED FOR THE VALUE 0 WHICH OCCURS WHEN IND1 IS THE FIRST CELL IN
C  THE STRING AA
      PREV=0
      IF(STRING-INDEX) 1,4,1
    1 PREV=STRING
    2 NPNTR=SPACE(PREV+1)
      IF(NPNTR-INDEX) 3,4,3
COND  EXISTS(((SPACE(PREV) .NE. ELT(II,L)),(SPACE(INDEX)=ELT(II+1,L))),
COND   II,1,LENGTH(L)),SPACE(PREV+1) .NE. INDEX,LIST.PIR(PREV,L),
COND   LIST.PIR(NPNTR,L)
    3 PREV=NPNTR
      GO TO 2
COND   ((SPACE(PREV+1)=INDEX),(EXISTS((SPACE(PREV)=ELT(II,L)),II,1,
COND   LENGTH(L)))) .OR. ((PREV=0),(STRING=INDEX))
    4 RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND EXISTS((SPACE(INDEX) .EQ. ELT(II+1,L)),II+1,1,LENGTH(L)) .AND.		
COND LV(STRING,L)		
INTEGER FUNCTION PREV(STRING,INDEX)		
INTEGER SPACE,STRING		
COMMON/BLOCK/SPACE(10000),IQQ,IQQQ	[PREV]	[]
PREV=0		
IF(STRING-INDEX .EQ. 0)GO TO 4	[PREV]	[STRING]
1 PREV=STRING	[NPNTR]	[SPACE,PREV]
2 NPNTR=SPACE(PREV+1)		
IF(NPNTR-INDEX .EQ. 0)GO TO 4		
COND EXISTS(((SPACE(PREV) .NE. ELT(II,L)),(SPACE(INDEX) .EQ.		
COND ELT(II+1,L))),II,1,LENGTH(L)) .AND. SPACE(PREV+1) .NE. INDEX .AND.		
COND LIST.PIR(PREV,L) .AND. LIST.PIR(NPNTR,L)	[PREV]	[NPNTR]
3 PREV=NPNTR		
GO TO 2		
COND ((SPACE(PREV+1) .EQ. INDEX),(EXISTS((SPACE(PREV) .EQ.		
COND ELT(II,L)),II,1,LENGTH(L)))) .OR. ((PREV .EQ. 0),(STRING .EQ.		
COND INDEX))		
4 RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND EXISTS((SPACE(INDEX) .EQ. ELT(II+1,L)),II+1,1,LENGTH(L)) .AND. LV(STRING,L) INTEGER FUNCTION PREV(STRING,INDEX) INTEGER SPACE,STRING COMMON/BLOCK/SPACE(10000),IQQ,IQQQ PREV=0 (STRING-INDEX .EQ. 0) COND ((SPACE(PREV+1) .EQ. INDEX),(EXISTS((SPACE(PREV) .EQ. COND ELT(II,L)),II,1,LENGTH(L)))) .OR. ((PREV .EQ. 0),(STRING .EQ. INDEX))	((EXISTS(0 .EQ. ELT(1+II,L)-SPACE(INDEX),1+II,1,LENGT H(L))) .AND. (LV(STRING,L)) .AND. (0 .EQ. -INDEX+STRING)) .IMPLIES. ((0 .EQ. INDEX-SPACE(1),EXISTS(0 .EQ. ELT(II,L)-SPACE(0),II,1,LENGTH(L))) .OR. (.TRUE. ,0 .EQ. INDEX-STRING))
2	COND EXISTS((SPACE(INDEX) .EQ. ELT(II+1,L)),II+1,1,LENGTH(L)) .AND. LV(STRING,L) INTEGER FUNCTION PREV(STRING,INDEX) INTEGER SPACE,STRING COMMON/BLOCK/SPACE(10000),IQQ,IQQQ PREV=0 .NOT. (STRING-INDEX .EQ. 0) 1 PREV=STRING 2 NPNTR=SPACE(PREV+1) (NPNTR-INDEX .EQ. 0) COND ((SPACE(PREV+1) .EQ. INDEX),(EXISTS((SPACE(PREV) .EQ. COND ELT(II,L)),II,1,LENGTH(L)))) .OR. ((PREV .EQ. 0),(STRING .EQ. INDEX))	((EXISTS(0 .EQ. ELT(1+II,L)-SPACE(INDEX),1+II,1,LENGT H(L))) .AND. (LV(STRING,L)) .AND. (0 .NE. -INDEX+STRING) .AND. (0 .EQ. -INDEX+SPACE(1+STRING))) .IMPLIES. ((0 .EQ. INDEX-SPACE(1+STRING),EXISTS(0 .EQ. ELT(II,L)-SPACE(STRING),II,1,LENGTH(L))) .OR. (0 .EQ. STRING,0 .EQ. INDEX-STRING))
3	COND EXISTS((SPACE(INDEX) .EQ. ELT(II+1,L)),II+1,1,LENGTH(L)) .AND. LV(STRING,L) INTEGER FUNCTION PREV(STRING,INDEX) INTEGER SPACE,STRING COMMON/BLOCK/SPACE(10000),IQQ,IQQQ PREV=0	

```
        .NOT. (STRING-INDEX .EQ. 0)                                                                    1
1       PREV=STRING
2       NPNTR=SPACE(PREV+1)
         .NOT. (NPNTR-INDEX .EQ. 0)
COND EXISTS(((SPACE(PREV) .NE. ELT(II,L)),(SPACE(INDEX) .EQ.
COND ELT(II+1,L))),II,1,LENGTH(L)) .AND. SPACE(PREV+1) .NE. INDEX .AND.
COND LIST.PIR(PREV,L) .AND. LIST.PIR(NPNTR,L)                                                          2
                                                                    ((EXISTS(0 .EQ.
                                                                    ELT(1+II,L)-SPACE(INDEX),1+II,1,LENGT
                                                                    H(L))) .AND. (LV(STRING,L)) .AND. (0
                                                                    .NE. -INDEX+STRING) .AND. (0 .NE.
                                                                    -INDEX+SPACE(1+STRING)))
                                                                     .IMPLIES.                         3
                                                                    ((LIST.PIR(STRING,L)) .AND.
                                                                    (LIST.PIR(SPACE(1+STRING),L)) .AND.
                                                                    (0 .NE. INDEX-SPACE(1+STRING)) .AND.
                                                                    (EXISTS(0 .NE.
                                                                    ELT(II,L)-SPACE(STRING),0 .EQ.
                                                                    ELT(1+II,L)-SPACE(INDEX),II,1,LENGTH(
                                                                    L))))

4
COND EXISTS(((SPACE(PREV) .NE. ELT(II,L)),(SPACE(INDEX) .EQ.
COND ELT(II+1,L))),II,1,LENGTH(L)) .AND. SPACE(PREV+1) .NE. INDEX .AND.
COND LIST.PIR(PREV,L) .AND. LIST.PIR(NPNTR,L)
3       PREV=NPNTR
2       NPNTR=SPACE(PREV+1)
        (NPNTR-INDEX .EQ. 0)
COND ((SPACE(PREV+1) .EQ. INDEX),(EXISTS((SPACE(PREV) .EQ.
COND ELT(II,L)),II,1,LENGTH(L)))) .OR. ((PREV .EQ. 0),(STRING .EQ. INDEX))
                                                                    ((EXISTS(0 .NE.
                                                                    ELT(II,L)-SPACE(PREV),0 .EQ.
                                                                    ELT(1+II,L)-SPACE(INDEX),II,1,LENGTH(
                                                                    L))) .AND. (0 .NE.
                                                                    INDEX-SPACE(1+PREV)) .AND.
                                                                    (LIST.PIR(PREV,L)) .AND.
                                                                    (LIST.PIR(NPNTR,L)) .AND. (0 .EQ.
                                                                    -INDEX+SPACE(1+NPNTR)))
                                                                     .IMPLIES.
                                                                    ((0 .EQ.
                                                                    INDEX-SPACE(1+NPNTR),EXISTS(0 .EQ.
                                                                    ELT(II,L)-SPACE(NPNTR),II,1,LENGTH(L)
                                                                    )) .OR. (0 .EQ. NPNTR,0 .EQ.
                                                                    INDEX-STRING))

5
COND EXISTS(((SPACE(PREV) .NE. ELT(II,L)),(SPACE(INDEX) .EQ.
COND ELT(II+1,L))),II,1,LENGTH(L)) .AND. SPACE(PREV+1) .NE. INDEX .AND.
COND LIST.PIR(PREV,L) .AND. LIST.PIR(NPNTR,L)
3       PREV=NPNTR
2       NPNTR=SPACE(PREV+1)
         .NOT. (NPNTR-INDEX .EQ. 0)
COND EXISTS(((SPACE(PREV) .NE. ELT(II,L)),(SPACE(INDEX) .EQ.
COND ELT(II+1,L))),II,1,LENGTH(L)) .AND. SPACE(PREV+1) .NE. INDEX .AND.
```

```
COND (LIST.PIR(PREV,L) .AND. LIST.PIR(NPNTR,L)
                                                ((EXISTS(0 .NE.
                                                ELT(II,L)-SPACE(PREV),0 .EQ.
                                                ELT(1+II,L)-SPACE(INDEX),II,1,LENGTH(
                                                L))) .AND. (0 .NE.
                                                INDEX-SPACE(1+PREV)) .AND.
                                                (LIST.PIR(PREV,L)) .AND.
                                                (LIST.PIR(NPNTR,L)) .AND. (0 .NE.
                                                -INDEX+SPACE(1+NPNTR)))
                                                  .IMPLIES.
                                                ((LIST.PIR(NPNTR,L)) .AND.
                                                (LIST.PIR(SPACE(1+NPNTR),L)) .AND.
                                                (0 .NE. INDEX-SPACE(1+NPNTR)) .AND.
                                                (EXISTS(0 .NE.
                                                ELT(II,L)-SPACE(NPNTR),0 .EQ.
                                                ELT(1+II,L)-SPACE(INDEX),II,1,LENGTH(
                                                L))))
```

2,24) PREV(STRING,INDEX)

This function returns a pointer to the character before the character pointed to by INDEX in the string STRING.

VERIFICATION:

Path 1: Since INDEX=STRING, the conclusion is true.

Path 2: INDEX=SPACE(STRING+1) is not changed. In addition, since EXISTS(ELT(II+1,L)=SPACE(INDEX),II+1,!, LENGTH(L)), therefore EXISTS(ELT(II,L)=SPACE(STRING),II,1,LENGTH(L)).

Path 3: INDEX≠SPACE(STRING+1) is unchanged in the path. Since EXISTS(ELT(II+1,L)=SPACE(INDEX),II+1,1, LENGTH(L)), hence EXISTS(ELT(II,L)≠SPACE(STRING), II,1,LENGTH(L)).

Path 4: This path is similar to path 2 with NPNTR replaces STRING.

Path 5: This path is similar to path 3.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC    .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC    LIST.PIR(P,L)=EXISTS(P=ADDR(K,L),K,1,LENGTH(L)).
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    ALL(F,X,I,J)=F(X), X .IN. (I..J).
CC    ALL(ALL(J,A,X,I,B,Y))=ALL(ALL(ELT(I,L1) .NE. ELT(J,L2),
CC    J,A,LENGTH(X)),I,B,LENGTH(Y)).
CC    SL=SUBLIST.
COND   ANC=0 .OR. ANC=1,
COND   LV(SLOCS,L1),LV(PLOCS,L2)
      SUBROUTINE SPS(SLOCS,PLOCS,BEGS,ENDS,ANC)
      COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
      INTEGER SLOCS,PLOCS,BEGS,ENDS,ANC,FAIL,SPACE,SP,PP,OLDSP
      BEGS = 0
      SP = SLOCS
      OLDSP = SLOCS
      IF(SLOCS) 1,10,1
    1 PP = PLOCS
    2 IF(SPACE(SP)-SPACE(PP)) 3,6,3
COND   ANC=0 .OR. ANC=1,
COND   ALL((SPACE(SP) .NE. ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,
COND   SPACE(PP+1),L2))),
COND   ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))),LIST.PIR(PP,L2)
    3 PP = SPACE(PP+1)
      IF(PP) 2,4,2
    4 IF(BEGS) 11,5,11
    5 IF(ANC) 10,8,10
    6 IF(BEGS) 8,7,8
    7 BEGS = SP
COND   ANC=0 .OR. ANC=1,
COND   (SPACE(SP)=SPACE(PP)) .OR.
COND   (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
    8 OLDSP = SP
      SP = SPACE(SP+1)
      IF(SP) 1,9,1
    9 IF(BEGS) 11,10,11
   10 FAIL = 1
      ENDS = OLDSP
COND   FAIL=1,((LV(SLOCS,L1),(L1=NIL)) .OR.
COND   (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.
COND   (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1)))))
      RETURN
   11 FAIL = 0
      ENDS = OLDSP
COND   FAIL=0,
COND   ((ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.
COND   (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1)))))
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND ANC=0 .OR. ANC=1 .AND. LV(SLOCS,L1) .AND. LV(PLOCS,L2)		
SUBROUTINE SPAN(SLOCS,PLOCS,BEGS,ENDS,ANC)		
COMMON/BLOCK/SPACE(10000),FAIL,IFQQ		
INTEGER SLOCS,PLOCS,BEGS,ENDS,ANC,FAIL,SPACE,SP,PP,OLDSP		
BEGS=0	[BEGS]	[]
SP=SLOCS	[SP]	[SLOCS]
OLDSP=SLOCS	[OLDSP]	[SLOCS]
IF(SLOCS .EQ. 0)GO TO 10		
1 PP=PLOCS	[PP]	[PLOCS]
2 IF(SPACE(SP)-SPACE(PP) .EQ. 0)GO TO 6		
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.		
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.		
COND ALL(ALL(J,1,L2,T,1,SL(SLOCS,SP,L1))) .AND. LIST.PTR(PP,L2)		
3 PP=SPACE(PP+1)	[PP]	[SPACE,PP]
IF(PP .NE. 0)GO TO 2		
4 IF(BEGS .NE. 0)GO TO 11		
5 IF(ANC .NE. 0)GO TO 10		
GO TO 8		
6 IF(BEGS .NE. 0)GO TO 8		
7 BEGS=SP	[BEGS]	[SP]
COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.		
COND (ALL(ALL(J,1,L2,T,1,SL(SLOCS,SP,L1))))		
8 OLDSP=SP	[OLDSP]	[SP]
SP=SPACE(SP+1)	[SP]	[SPACE,SP]
IF(SP .NE. 0)GO TO 1		
9 IF(BEGS .NE. 0)GO TO 11		
10 FAIL=1	[FAIL]	[]
ENDS=OLDSP	[ENDS]	[OLDSP]
COND FAIL=1 .AND. ((LV(SLOCS,L1),(L1 .EQ. NIL)) .OR.		
COND (ALL(ALL(J,1,L2,T,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.		
COND (ALL(ALL(J,1,L2,T,1,SL(SLOCS,OLDSP,L1)))))		
RETURN		
11 FAIL=0	[FAIL]	[]
ENDS=OLDSP	[ENDS]	[OLDSP]
COND FAIL=0 .AND. ((ALL(ALL(J,1,L2,T,1,SL(SLOCS,SPACE(SP+1),L1))) .OR.		
COND (ALL(ALL(J,1,L2,T,1,SL(SLOCS,OLDSP,L1))))		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND ANC=0 .OR. ANC=1 .AND. LV(SLOCS,L1) .AND. LV(PLOCS,L2) SUBROUTINE SPAN(SLOCS,PLOCS,BEGS,ENDS,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCS,PLOCS,BEGS,ENDS,ANC,FAIL,SPACE,SP,PP,OLDSP BEGS=0 SP=SLOCS OLDSP=SLOCS (SLOCS .EQ. 0) 10 FAIL=1 ENDS=OLDSP COND FAIL=1 .AND. ((LV(SLOCS,L1),(L1 .EQ. NIL)) .OR. COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR. COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1))))	(((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC) .AND. (LV(SLOCS,L1)) .AND. (LV(PLOCS,L2)))) .AND. (0 .EQ. SLOCS)) .IMPLIES. (((LV(SLOCS,L1),0 .EQ. -L1+NIL) .OR. (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1+SLOCS),L1)))) .OR. (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SLOCS,L1))))
2	COND ANC=0 .OR. ANC=1 .AND. LV(SLOCS,L1) .AND. LV(PLOCS,L2) SUBROUTINE SPAN(SLOCS,PLOCS,BEGS,ENDS,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCS,PLOCS,BEGS,ENDS,ANC,FAIL,SPACE,SP,PP,OLDSP BEGS=0 SP=SLOCS OLDSP=SLOCS .NOT. (SLOCS .EQ. 0) 1 PP=PLOCS 2 (SPACE(SP)-SPACE(PP) .EQ. 0) 6 (BEGS .NE. 0)	
3	COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR. COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))	NONE
2	3	
1	COND ANC=0 .OR. ANC=1 .AND. LV(SLOCS,L1) .AND. LV(PLOCS,L2) SUBROUTINE SPAN(SLOCS,PLOCS,BEGS,ENDS,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCS,PLOCS,BEGS,ENDS,ANC,FAIL,SPACE,SP,PP,OLDSP BEGS=0 SP=SLOCS OLDSP=SLOCS	

```
        .NOT. (SLOCS .EQ. 0)
1       PP=PLOCS
2       (SPACE(SP)-SPACE(PP) .EQ. 0).
6        .NOT. (BEGS .NE. 0)
7       BEGS=SP
COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
                                                        (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                        .AND. (LV(SLOCS,L1)) .AND.
                                                        (LV(PLOCS,L2)))) .AND. (0 .NE.
                                                        SLOCS) .AND. (0 .EQ.
                                                        -SPACE(PLOCS)+SPACE(SLOCS)))
                                                         .IMPLIES.
                                                        (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                        .AND. (0 .EQ.
                                                        SPACE(PLOCS)-SPACE(SLOCS)))) .OR.
                                                        (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SLOCS,L1
                                                        )))))

4
COND ANC=0 .OR. ANC=1 .AND. LV(SLOCS,L1) .AND. LV(PLOCS,L2)
        SUBROUTINE SPAN(SLOCS,PLOCS,BEGS,ENDS,ANC)
        COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
        INTEGER SLOCS,PLOCS,BEGS,ENDS,ANC,FAIL,SPACE,SP,PP,OLDSP
        BEGS=0
        SP=SLOCS
        OLDSP=SLOCS
         .NOT. (SLOCS .EQ. 0)
1       PP=PLOCS
2        .NOT. (SPACE(SP)-SPACE(PP) .EQ. 0)
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
                                                        (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                        .AND. (LV(SLOCS,L1)) .AND.
                                                        (LV(PLOCS,L2)))) .AND. (0 .NE.
                                                        SLOCS) .AND. (0 .NE.
                                                        -SPACE(PLOCS)+SPACE(SLOCS)))
                                                         .IMPLIES.
                                                        ((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                        .AND. (ALL(0 .NE.
                                                        ELT(J,L2)-SPACE(SLOCS),J,1,LENGTH(SUB
                                                        LIST(PLOCS,SPACE(1+PLOCS),L2))))
                                                        .AND.
                                                        (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SLOCS,L1
                                                        )))) .AND. (LIST.PIR(PLOCS,L2))))

5
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
3       PP=SPACE(PP+1)
        (PP .NE. 0)
2       (SPACE(SP)-SPACE(PP) .EQ. 0)
```

```
6     (BEGS .NE. 0)
COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
                                                    (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCS,SPACE(1+PP),L2)))) .AND.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ) .AND. (LIST.PIR(PP,L2)))) .AND. (0
                                                    .NE. -SPACE(1+PP)) .AND. (0 .EQ.
                                                    -SPACE(SPACE(1+PP))+SPACE(SP)) .AND.
                                                    (0 .NE. -BEGS))
                                                     .IMPLIES.
                                                    (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (0 .EQ.
                                                    SPACE(SPACE(1+PP))-SPACE(SP)))) .OR.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ))

6
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
3     SP=SPACE(PP+1)
      (PP .NE. 0)
2     (SPACE(SP)-SPACE(PP) .EQ. 0)
6      .NOT. (BEGS .NE. 0)
7     BEGS=SP
COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
                                                    (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCS,SPACE(1+PP),L2)))) .AND.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ) .AND. (LIST.PIR(PP,L2)))) .AND. (0
                                                    .NE. -SPACE(1+PP)) .AND. (0 .EQ.
                                                    -SPACE(SPACE(1+PP))+SPACE(SP)) .AND.
                                                    (0 .EQ. -BEGS))
                                                     .IMPLIES.
                                                    (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (0 .EQ.
                                                    SPACE(SPACE(1+PP))-SPACE(SP)))) .OR.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ))

7
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
3     SP=SPACE(PP+1)
      (PP .NE. 0)
2      .NOT. (SPACE(SP)-SPACE(PP) .EQ. 0)
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
```

```
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
                                                    (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCS,SPACE(1+PP),L2)))) .AND.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ) .AND. (LIST.PIR(PP,L2)))) .AND. (0
                                                    .NE. -SPACE(1+PP)) .AND. (0 .NE.
                                                    -SPACE(SPACE(1+PP))+SPACE(SP)))
                                                     .IMPLIES.
                                                    ((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCS,SPACE(1+SPACE(1+PP)),L2))))
                                                    .AND.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ) .AND. (LIST.PIR(SPACE(1+PP),L2))))

8
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
3       PP=SPACE(PP+1)
         .NOT. (PP .NE. 0)
4       (BEGS .NE. 0)
11      FAIL=0
        ENDS=OLDSP
COND FAIL=0 .AND. ((ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.
COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1)))))
                                                    (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                    .AND. (ALL(0 .NE.
                                                    ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                    T(PLOCS,SPACE(1+PP),L2)))) .AND.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                    ) .AND. (LIST.PIR(PP,L2)))) .AND. (0
                                                    .EQ. -SPACE(1+PP)) .AND. (0 .NE.
                                                    -BEGS))
                                                     .IMPLIES.
                                                    ((ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1
                                                    +SP),L1)))) .OR.
                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1
                                                    )))))

9
COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
3       PP=SPACE(PP+1)
         .NOT. (PP .NE. 0)
4        .NOT. (BEGS .NE. 0)
5       (ANC .NE. 0)
10      FAIL=1
        ENDS=OLDSP
```

```
      COND FAIL=1 .AND. ((LV(SLOCS,L1),(L1 .EQ. NIL)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1))))))
                                                          (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                          .AND. (ALL(0 .NE.
                                                          ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                          T(PLOCS,SPACE(1+PP),L2)))) .AND.
                                                          (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                          ) .AND. (LIST.PIR(PP,L2)))) .AND. (0
                                                          .EQ. -SPACE(1+PP)) .AND. (0 .EQ.
                                                          -BEGS) .AND. (0 .NE. -ANC))
                                                           .IMPLIES.
                                                          (((LV(SLOCS,L1),0 .EQ. -L1+NIL) .OR.
                                                          (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1+
                                                          SP),L1))))) .OR.
                                                          (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1
                                                          ))))))

10
      COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
      COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
      3     PP=SPACE(PP+1)
             .NOT. (PP .NE. 0)
      4      .NOT. (BEGS .NE. 0)
      5      .NOT. (ANC .NE. 0)
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
                                                          (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                          .AND. (ALL(0 .NE.
                                                          ELT(J,L2)-SPACE(SP),J,1,LENGTH(SUBLIS
                                                          T(PLOCS,SPACE(1+PP),L2)))) .AND.
                                                          (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                          ) .AND. (LIST.PIR(PP,L2)))) .AND. (0
                                                          .EQ. -SPACE(1+PP)) .AND. (0 .EQ.
                                                          -BEGS) .AND. (0 .EQ. -ANC))
                                                           .IMPLIES.
                                                          (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                          .AND. (0 .EQ.
                                                          SPACE(SPACE(1+PP))-SPACE(SP)))) .OR.
                                                          (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                          ))

11
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
      9     OLDSP=SP
             SP=SPACE(SP+1)
             (SP .NE. 0)
      1      PP=PLOCS
      2      (SPACE(SP)-SPACE(PP) .EQ. 0)
      6      (BEGS .NE. 0)
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
```

```
                                                                (((((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)                    1
                                                                .AND. (0 .EQ. SPACE(PP)-SPACE(SP))))
                                                                .OR.
                                                                (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                                )) .AND. (0 .NE. -SPACE(1+SP)) .AND.
                                                                (0 .EQ.
                                                                -SPACE(PLOCS)+SPACE(SPACE(1+SP)))                         2
                                                                .AND. (0 .NE. -BEGS))
                                                                 .IMPLIES.
                                                                (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                .AND. (0 .EQ.
                                                                SPACE(PLOCS)-SPACE(SPACE(1+SP)))))
                                                                .OR.                                                      3
                                                                (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1+
                                                                SP),L1)))))

12
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
      8     OLDSP=SP
            SP=SPACE(SP+1)
            (SP .NE. 0)
      1     PP=PLOCS
      2     (SPACE(SP)-SPACE(PP) .EQ. 0)
      6      .NOT. (BEGS .NE. 0)
      7     BEGS=SP
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
                                                                (((((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                .AND. (0 .EQ. SPACE(PP)-SPACE(SP))))
                                                                .OR.
                                                                (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                                )) .AND. (0 .NE. -SPACE(1+SP)) .AND.
                                                                (0 .EQ.
                                                                -SPACE(PLOCS)+SPACE(SPACE(1+SP)))
                                                                .AND. (0 .EQ. -BEGS))
                                                                 .IMPLIES.
                                                                (((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                .AND. (0 .EQ.
                                                                SPACE(PLOCS)-SPACE(SPACE(1+SP)))))
                                                                .OR.
                                                                (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1+
                                                                SP),L1)))))

13
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
      8     OLDSP=SP
            SP=SPACE(SP+1)
            (SP .NE. 0)
      1     PP=PLOCS
      2      .NOT. (SPACE(SP)-SPACE(PP) .EQ. 0)
      COND ANC=0 .OR. ANC=1 .AND. ALL((SPACE(SP) .NE.
      COND ELT(J,L2)),J,1,LENGTH(SUBLIST(PLOCS,SPACE(PP+1),L2))) .AND.
```

```
      COND ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) .AND. LIST.PIR(PP,L2)
                                                                    ((((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                    .AND. (0 .EQ. SPACE(PP)-SPACE(SP))))
                                                                    .OR.
                                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                                    )) .AND. (0 .NE. -SPACE(1+SP)) .AND.
                                                                    (0 .NE.
                                                                    -SPACE(PLOCS)+SPACE(SPACE(1+SP))))
                                                                     .IMPLIES.
                                                                    ((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                    .AND. (ALL(0 .NE.
                                                                    ELT(J,L2)-SPACE(SPACE(1+SP)),J,1,LENG
                                                                    TH(SUBLIST(PLOCS,SPACE(1+PLOCS),L2)))
                                                                    ) .AND.
                                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1+
                                                                    SP),L1)))) .AND.
                                                                    (LIST.PIR(PLOCS,L2))))
14
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
8     OLDSP=SP
      SP=SPACE(SP+1)
       .AND. (SP .NE. 0)
9     (BEGS .NE. 0)
11    FAIL=0
      ENDS=OLDSP
      COND FAIL=0 .AND. ((ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1)))))
                                                                    ((((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
                                                                    .AND. (0 .EQ. SPACE(PP)-SPACE(SP))))
                                                                    .OR.
                                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                                    )) .AND. (0 .EQ. -SPACE(1+SP)) .AND.
                                                                    (0 .NE. -BEGS))
                                                                     .IMPLIES.
                                                                    ((ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(1
                                                                    +SPACE(1+SP)),L1)))) .OR.
                                                                    (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1)))
                                                                    ))
15
      COND ANC=0 .OR. ANC=1 .AND. (SPACE(SP) .EQ. SPACE(PP)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))))
9     OLDSP=SP
      SP=SPACE(SP+1)
       .AND. (SP .NE. 0)
9      .AND. (BEGS .NE. 0)
10    FAIL=1
      ENDS=OLDSP
      COND FAIL=1 .AND. ((LV(SLOCS,L1),(L1 .EQ. NIL)) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1)))) .OR.
      COND (ALL(ALL(J,1,L2,I,1,SL(SLOCS,OLDSP,L1)))))
                                                                    ((((0 .EQ. ANC) .OR. ((0 .EQ. 1-ANC)
```

2.25) SPAN(SLOCS,PLOCS,BEGS,ENDS,ANC)

SLOCS=subject string
PLOCS=pattern string

BEGS and ENDS is set to the beginning and end respectively of the matched substring. If match is sucessful, BEGS and ENDS are set and FAIL is set to 0,otherwise, FAIL is set to1.

VERIFICATION:

Path 1: Since SLOCS=0, hence L1=nil.

Path 2: proved.

Path 3: Since SL(SLOCS,SLOCS,L1), SL1=nil.
SPACE(PLOCS)=SPACE(SLOCS) is not changed.

Path 4: Again SL1=nil. Since SPACE(SLOCS)≠SPACE(PLOCS), therefore ALL(SPACE(SLOCS)≠ELT(J,L2),J,1, LENGTH(SUBLIST(PLOCS,SPACE(PLOCS+1),L2))).

Path 5: SPACE(SP)=SPACE(SPACE(PP+1)) is not changed.
ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) is also preserved,

Path 6: Similar t o path 5.

Path 7: Since all conditions are preserved, the conclusion is true.

Path 8: Since ALL(SPACE(SP)≠ELT(J,L2),J,1,LENGTH(SUBLIST (PLOCS,SPACE(PP+1),L2))) and SPACE(PP+1)=0 and ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))), hence ALL(ALL(J,1,L2,I,1,SL(SLOCS,SPACE(SP+1),L1))).

Path 9: Similar to path 8.

Path 10: ALL(ALL(J,1,L2,I,1,SL(SLOCS,SP,L1))) is not changed.

Path 11: SPACE(PLOCS)=SPACE(SPACE(SP+1)) is not changed.

Path 12: Similar to path 11.

Path 13: Since SPACE(PLOCS)≠SPACE(SPACE(SP+1)) hence ALL(SPACE(SPACE(SP+1))≠ELT(J,L2),J,1,LENGTH (SUBLIST(PLOCS,SPACE(PLOCS+1),L2))).

Path 14: ALL(ALL(J,1,I2,I,1,SI(SLOCS,SP,I1))) has not been charged.

Path 15: Similar to path 14.

```
CC    LV(LP,L)=(IF L=NIL THEN LP=0, ELSE LP IS ODD AND
CC    SPACE(LP)=CAR(L), LV(SPACE(LP+1), CDR(L)).
CC    SUBLIST(J,K,L)=SL .ST. (IF J=K THEN SL=NIL ELSE (CAR(SL)=SPACE(J)
CC    .AND. CDR(SL)=SUBLIST(SPACE(J+1),K,L)))
CC    LIST.PIR(P,L)=EXISTS(P=ADDP(K,L),K,1,LENGTH(L)).
CC    LENGTH(L)=(IF L=NIL THEN 0 ELSE (LENGTH(L)=LENGTH(CDR(L))+1).
CC    CDRN(0,L)=L.
CC    CDRN(I,L)=CDR(CDRN(I-1,L)).
CC    ELT(I,L)=CAR(CDRN(I-1,L)).
CC    EXISTS(F,X,I,J)=F(X), I .LE. X, X .LE. J.
CC    EXISTS(F,(X,I,J),(Y,K,L))=F(X,Y), I .LE. X, X .LE. J, K .LE. Y,
CC    Y .LE. L.
CC    POINT(L)=(I...J), SPACE(I)=CAR(L), SPACE(SPACE(I+1))=ELT(2,L),...
CC    ...SPACE(J)=ELT(LENGTH(L),L).
COND   LV(SLOCS,L1),LV(PLOCST,L2),SUBLIST(BEGST,SPACE(ENDS+1),L1)=NIL
       SUBROUTINE STRING(SLOCST,PLOCST,BEGST,ENDST,ANC)
      COMMON/BLOCK/SPACE(10000),FAIL,IFQQ
      INTEGER SLOCST,PLOCST,BEGST,ENDST,ANC,SPACE,FAIL
      IF(SLOCST.EQ.0)GO TO 5
      K=SLOCST
2      IF(SPACE(K).EQ.SPACE(PLOCST))GO TO 1
COND   SPACE(I) .NE. SPACE(J),LIST.PIR(I,L1),LIST.PIR(J,L2)
4      IF(ANC.EQ.1)GO TO 5
      K=SPACE(K+1)
      IF(K.NE.0)GO TO 2
5      FAIL=1
COND   FAIL=1
      RETURN
COND   SPACE(K)=CAR(L2)
1      KK=K
      L=PLOCST
COND   EXISTS((ELT(I,L1)=ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),
COND   L1))),(J,1,LENGTH(SUBLIST(PLOCST,SPACE(L+1),L2)))),
COND   LIST.PIR(KK,L1),LIST.PIR(L,L2)
6      L=SPACE(L+1)
      IF(L.EQ.0)GO TO 3
      KK=SPACE(KK+1)
      IF(KK.EQ.0)GO TO 5
      IF(SPACE(KK)-SPACE(L))4,6,4
3      BEGST=K
      ENDST=KK
      FAIL=0
COND   SUBLIST(BEGST,SPACE(ENDST+1),L1),LV(PLOCST,L2),FAIL=0,L2=SL1
      RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(SLOCS,L1) .AND. LV(PLOCST,L2) .AND.		
COND SUBLIST(BEGST,SPACE(ENDS+1),L1)=NIL		
SUBROUTINE STRING(SLOCST,PLOCST,BEGST,ENDST,ANC)		
COMMON/BLOCK/SPACE(10000),FAIL,IFQQ		
INTEGER SLOCST,PLOCST,BEGST,ENDST,ANC,SPACE,FAIL		
IF(SLOCST .EQ. 0)GO TO 5		
K=SLOCST	[K]	[SLOCST]
2 IF(SPACE(K) .EQ. SPACE(PLOCST))GO TO 1		
COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)		
4 IF(ANC .EQ. 1)GO TO 5		
K=SPACE(K+1)	[K]	[SPACE,K]
IF(K .NE. 0)GO TO 2		
5 FAIL=1	[FAIL]	[]
COND FAIL=1		
RETURN		
COND SPACE(K)=CAR(L2)		
1 KK=K	[KK]	[K]
L=PLOCST	[L]	[PLOCST]
COND EXISTS((ELT(I,L1) .E).		
COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBL		
COND IST(PLOCST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND.		
COND LIST.PIR(L,L2)		
6 L=SPACE(L+1)	[L]	[SPACE,L]
IF(L .EQ. 0)GO TO 3		
KK=SPACE(KK+1)	[KK]	[SPACE,KK]
IF(KK .EQ. 0)GO TO 5		
IF(SPACE(KK)-SPACE(L) .NE. 0)GO TO 4		
GO TO 6		
3 BEGST=K	[BEGST]	[K]
ENDST=KK	[ENDST]	[KK]
FAIL=0	[FAIL]	[]
COND SUBLIST(BEGST,SPACE(ENDST+1),L1) .AND. LV(PLOCST,L2) .AND. FAIL=0		
COND .AND. L2=SL1		
RETURN		
END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(SLOCS,L1) .AND. LV(PLOCST,L2) .AND. SUBLIST(BEGST,SPACE(ENDS+1),L1)=NIL SUBROUTINE STRING(SLOCST,PLOCST,BEGST,ENDST,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCST,PLOCST,BEGST,ENDST,ANC,SPACE,FAIL (SLOCST .EQ. 0) 5 FAIL=1 COND FAIL=1	NONE
2	COND LV(SLOCS,L1) .AND. LV(PLOCST,L2) .AND. SUBLIST(BEGST,SPACE(ENDS+1),L1)=NIL SUBROUTINE STRING(SLOCST,PLOCST,BEGST,ENDST,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCST,PLOCST,BEGST,ENDST,ANC,SPACE,FAIL .NOT. (SLOCST .EQ. 0) K=SLOCST 2 (SPACE(K) .EQ. SPACE(PLOCST)) COND SPACE(K)=CAR(L2)	((LV(SLOCS,L1)) .AND. (LV(PLOCST,L2)) .AND. (0 .EQ. NIL-SUBLIST(BEGST,SPACE(1+ENDS),L1)) .AND. (0 .NE. SLOCST) .AND. (0 .EQ. SPACE(PLOCST)-SPACE(SLOCST))) .IMPLIES. ((0 .EQ. CAR(L2)-SPACE(SLOCST)))
3	COND LV(SLOCS,L1) .AND. LV(PLOCST,L2) .AND. SUBLIST(BEGST,SPACE(ENDS+1),L1)=NIL SUBROUTINE STRING(SLOCST,PLOCST,BEGST,ENDST,ANC) COMMON/BLOCK/SPACE(10000),FAIL,IFQQ INTEGER SLOCST,PLOCST,BEGST,ENDST,ANC,SPACE,FAIL .NOT. (SLOCST .EQ. 0) K=SLOCST 2 .NOT. (SPACE(K) .EQ. SPACE(PLOCST)) COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)	((LV(SLOCS,L1)) .AND. (LV(PLOCST,L2)) .AND. (0 .EQ. NIL-SUBLIST(BEGST,SPACE(1+ENDS),L1)) .AND. (0 .NE. SLOCST) .AND. (0 .NE. SPACE(PLOCST)-SPACE(SLOCST))) .IMPLIES. ((LIST.PIR(I,L1)) .AND. (LIST.PIR(J,L2)) .AND. (0 .NE. -SPACE(I)+SPACE(J)))
4	COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)	

```
4       (ANC .EQ. 1)                                                    1
5       FAIL=1
COND FAIL=1                                                     NONE


5                                                                       2
COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)
4       .NOT. (ANC .EQ. 1)
        K=SPACE(K+1)
        (K .NE. 0)
2       (SPACE(K) .EQ. SPACE(PLOCST))
COND SPACE(K)=CAR(L2)                                                   3
                                                        ((0 .NE. -SPACE(I)+SPACE(J)) .AND.
                                                        (LIST.PIR(I,L1)) .AND.
                                                        (LIST.PIR(J,L2)) .AND. (0 .NE.
                                                        1-ANC) .AND. (0 .NE. -SPACE(1+K))
                                                        .AND. (0 .EQ.
                                                        SPACE(PLOCST)-SPACE(SPACE(1+K))))
                                                         .IMPLIES.
                                                        ((0 .EQ. CAR(L2)-SPACE(SPACE(1+K))))


6
COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)
4       .NOT. (ANC .EQ. 1)
        K=SPACE(K+1)
        (K .NE. 0)
2       .NOT. (SPACE(K) .EQ. SPACE(PLOCST))
COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)
                                                        ((0 .NE. -SPACE(I)+SPACE(J)) .AND.
                                                        (LIST.PIR(I,L1)) .AND.
                                                        (LIST.PIR(J,L2)) .AND. (0 .NE.
                                                        1-ANC) .AND. (0 .NE. -SPACE(1+K))
                                                        .AND. (0 .NE.
                                                        SPACE(PLOCST)-SPACE(SPACE(1+K))))
                                                         .IMPLIES.
                                                        ((LIST.PIR(I,L1)) .AND.
                                                        (LIST.PIR(J,L2)) .AND. (0 .NE.
                                                        -SPACE(I)+SPACE(J)))


7
COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)
4       .NOT. (ANC .EQ. 1)
        K=SPACE(K+1)
         .NOT. (K .NE. 0)
5       FAIL=1
COND FAIL=1                                                     NONE


8
COND SPACE(K)=CAR(L2)
1       KK=K
        L=PLOCST
```

```
      COND EXISTS((ELT(I,L1) .EQ.
      COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBLIST(PLOC
      COND ST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND. LIST.PIR(L,L2)
                                                                  ((0 .EQ. CAR(L2)-SPACE(K)))
                                                                   .IMPLIES.
                                                                  ((LIST.PIR(K,L1)) .AND.
                                                                  (LIST.PIR(PLOCST,L2)) .AND.
                                                                  (EXISTS(0 .EQ.
                                                                  -ELT(I,L1)+ELT(J,L2);I,1,LENGTH(SUBLI
                                                                  ST(K,SPACE(1+K),L1)),J,1,LENGTH(SUBLI
                                                                  ST(PLOCST,SPACE(1+PLOCST),L2)))))

    9
      COND EXISTS((ELT(I,L1) .EQ.
      COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBLIST(PLOC
      COND ST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND. LIST.PIR(L,L2)
      6     L=SPACE(L+1)
            (L .EQ. 0)
      3     BEGST=K
            ENDST=KK
            FAIL=0
      COND SUBLIST(BEGST,SPACE(ENDST+1),L1) .AND. LV(PLOCST,L2) .AND. FAIL=0 .AND.
      COND L2=SL1
                                                                  ((EXISTS(0 .EQ.
                                                                  -ELT(I,L1)+ELT(J,L2),I,1,LENGTH(SUBLI
                                                                  ST(K,SPACE(1+KK),L1)),J,1,LENGTH(SUBL
                                                                  IST(PLOCST,SPACE(1+L),L2)))) .AND.
                                                                  (LIST.PIR(KK,L1)) .AND.
                                                                  (LIST.PIR(L,L2)) .AND. (0 .EQ.
                                                                  SPACE(1+L)))
                                                                   .IMPLIES.
                                                                  ((SUBLIST(K,SPACE(1+KK),L1)) .AND.
                                                                  (LV(PLOCST,L2)) .AND. (0 .EQ.
                                                                  -L2+SL1))

   10
      COND EXISTS((ELT(I,L1) .EQ.
      COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBLIST(PLOC
      COND ST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND. LIST.PIR(L,L2)
      6     L=SPACE(L+1)
             .NOT. (L .EQ. 0)
      3     KK=SPACE(KK+1)
            (KK .EQ. 0)
      5     FAIL=1
      COND FAIL=1
                                                                  NONE

   11
      COND EXISTS((ELT(I,L1) .EQ.
      COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBLIST(PLOC
      COND ST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND. LIST.PIR(L,L2)
      6     L=SPACE(L+1)
             .NOT. (L .EQ. 0)
```

```
      KK=SPACE(KK+1)
       .NOT. (KK .EQ. 0)
       (SPACE(KK)-SPACE(L) .NE. 0)
COND SPACE(I) .NE. SPACE(J) .AND. LIST.PIR(I,L1) .AND. LIST.PIR(J,L2)
                                                    ((EXISTS(0 .EQ.
                                                    -ELT(I,L1)+ELT(J,L2),I,1,LENGTH(SUBLI
                                                    ST(K,SPACE(1+KK),L1)),J,1,LENGTH(SUBL
                                                    IST(PLOCST,SPACE(1+L),L2)))) .AND.
                                                    (LIST.PIR(KK,L1)) .AND.
                                                    (LIST.PIR(L,L2)) .AND. (0 .NE.
                                                    SPACE(1+L)) .AND. (0 .NE.
                                                    SPACE(1+KK)) .AND. (0 .NE.
                                                    -SPACE(SPACE(1+KK))+SPACE(SPACE(1+L))
                                                    ))
                                                     .IMPLIES.
                                                    ((LIST.PIR(I,L1)) .AND.
                                                    (LIST.PIR(J,L2)) .AND. (0 .NE.
                                                    -SPACE(I)+SPACE(J)))

12
COND EXISTS((ELT(I,L1) .EQ.
COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBLIST(PLOC
COND ST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND. LIST.PIR(L,L2)
6     L=SPACE(L+1)
       .NOT. (L .EQ. 0)
      KK=SPACE(KK+1)
       .NOT. (KK .EQ. 0)
       .NOT. (SPACE(KK)-SPACE(L) .NE. 0)
COND EXISTS((ELT(I,L1) .EQ.
COND ELT(J,L2)),(I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1))),(J,1,LENGTH(SUBLIST(PLOC
COND ST,SPACE(L+1),L2)))) .AND. LIST.PIR(KK,L1) .AND. LIST.PIR(L,L2)
                                                    ((EXISTS(0 .EQ.
                                                    -ELT(I,L1)+ELT(J,L2),I,1,LENGTH(SUBLI
                                                    ST(K,SPACE(1+KK),L1)),J,1,LENGTH(SUBL
                                                    IST(PLOCST,SPACE(1+L),L2)))) .AND.
                                                    (LIST.PIR(KK,L1)) .AND.
                                                    (LIST.PIR(L,L2)) .AND. (0 .NE.
                                                    SPACE(1+L)) .AND. (0 .NE.
                                                    SPACE(1+KK)) .AND. (0 .EQ.
                                                    -SPACE(SPACE(1+KK))+SPACE(SPACE(1+L))
                                                    ))
                                                     .IMPLIES.
                                                    ((LIST.PIR(SPACE(1+KK),L1)) .AND.
                                                    (LIST.PIR(SPACE(1+L),L2)) .AND.
                                                    (EXISTS(0 .EQ.
                                                    -ELT(I,L1)+ELT(J,L2),I,1,LENGTH(SUBLI
                                                    ST(K,SPACE(1+SPACE(1+KK)),L1)),J,1,LE
                                                    NGTH(SUBLIST(PLOCST,SPACE(1+SPACE(1+L
                                                    )),L2)))))
```

2.23) STRING(SLOCST,PLOCST,BEGST,ENDST,ANC)

SLOCST=subject string
PLOCST=pattern string.

If the pattern string is a substring of the subject string then a match is made, in unanchored mode. BEGST and ENDST is set respectively point to the beginning and the ending characters of the leftmost match in the subject string. If ANC=1, anchored mode then the matched substring must be an initial substring of the subject string.

VERIFICATION:

Path 1: proved.

Path 2: Since LV(PLOCST,L2) and SPACE(SLOCST)=SPACE(PLOCST), hence SPACE(SLOCST)=CAR(L2).

Path 3: insufficient assertion.

Path 4: proved.

Path 5: same as path 2.

Path 6: Since LIST.PIR(I,L1) and LIST.PIR(J,L2) and SPACE(SPACE(K+1))≠SPACE(PLOCST), hence SPACE(I)≠SPACE(J) is true.

Path 7: proved.

Path 8: insufficient assertion

Path 9: insufficient assertion

Path 10: proved.

Path 11: insufficient assertion

Path 12: Since EXISTS(EIT(I,L1)=EIT(J,L2),I,1,LENGTH(SUBLIST(K,SPACE(KK+1),L1)),J,1,LENGTH(SUBLIST(PLOCST,SPACE(L+1),L2))) and SPACE(SPACE(J+1))=SPACE(SPACE(KK+1)), hence EXISTS(EIT(I,L1)=EIT(J,L2),I,1,LENGTH(SUBLIST(K,SPACE(SPACE(KK+1)+1),L1)),J,1,LENGTH(SUBLIST(PLOCST,SPACE(SPACE(L+1)+1),L2)).

```
CC  LENGTH(L) = (IF NIL THEN 0 ELSE LENGTH(CDR(L))+1)
CC  ALL(F(X),X,I,L) .IN. (F(X), X.IN.(I..J))
CC  EXISTS(F(X),X,I,J) .IN. (F(X),I.LE.X,X.LE.J)
C   COMP(L2,L3) IS THE COMPOSITION OF LISTS L2 AND L3
CC  L=COMP(L2,L3) .IN. (ALL(ELT(K,L)=ELT(K,L2),K,1,LENGTH(L2)).OR.
CC  ALL(ELT(K,L)=ELT(K,L3),K,LENGTH(L2),LENGTH(LL)), LENGTH(L)=
CC  LENGTH(L2)+LENGTH(L3))
COND  LV(I1,L1), LV(I2,L2), LV(I3,L3), LV(I4,L4), LV(I5,L5),
COND  LV(I6,L6), LV(I7,L7), LV(I8,L8)
      SUBROUTINE CON(I1,I2,I3,I4,I5,I6,I7,I8)
      CALL CONCAT(I1,I1,I2)
      CALL CONCAT(I1,I1,I3)
      CALL CONCAT(I1,I1,I4)
      CALL CONCAT(I1,I1,I5)
      IF (I6.EQ.0) GO TO 1
      CALL CONCAT(I1,I1,I6)
      CALL CONCAT(I1,I1,I7)
      CALL CONCAT(I1,I1,I8)
COND  LV(I1,L) ,    L= COMP(COMP(COMP(COMP(COMP(COMP(COMP(L1,L2),L3),
COND  L4),L5),L6),L7),L8)
1     RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN

```
      COND LV(I1,L1) .AND. LV(I2,L2) .AND. LV(I3,L3) .AND. LV(I4,L4) .AND.
      COND LV(I5,L5) .AND. LV(I6,L6) .AND. LV(I7,L7) .AND. LV(I8,L8)
            SUBROUTINE CON(I1,I2,I3,I4,I5,I6,I7,I8)
            CALL CONCAT(I1,I1,I2)
            CALL CONCAT(I1,I1,I3)
            CALL CONCAT(I1,I1,I4)
            CALL CONCAT(I1,I1,I5)
            IF(I6 .EQ. 0)GO TO 1
            CALL CONCAT(I1,I1,I6)
            CALL CONCAT(I1,I1,I7)
            CALL CONCAT(I1,I1,I8)
      COND LV(I1,L1) .AND.
      COND L=COMP(COMP(COMP(COMP(COMP(COMP(COMP(L1,L2),L3),L4),L5),L6),L7),L8)
      1     RETURN
            END
```

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(I1,L1) .AND. LV(I2,L2) .AND. LV(I3,L3) .AND. LV(I4,L4) .AND. LV(I5,L5) COND .AND. LV(I6,L6) .AND. LV(I7,L7) .AND. LV(I8,L8) SUBROUTINE CON(I1,I2,I3,I4,I5,I6,I7,I8) CALL CONCAT(I1,I1,I2) CALL CONCAT(I1,I1,I3) CALL CONCAT(I1,I1,I4) CALL CONCAT(I1,I1,I5) (I6 .EQ. 0) COND LV(I1,L) .AND. COND L=COMP(COMP(COMP(COMP(COMP(COMP(COMP(L1,L2),L3),L4),L5),L6),L7),L8)	((LV(I1,L1)) .AND. (LV(I2,L2)) .AND. (LV(I3,L3)) .AND. (LV(I4,L4)) .AND. (LV(I5,L5)) .AND. (LV(I6,L6)) .AND. (LV(I7,L7)) .AND. (LV(I8,L8)) .AND. (0 .EQ. I6)) .IMPLIES. ((LV(I1,L)) .AND. (0 .EQ. COMP(COMP(COMP(COMP(COMP(COMP(COMP(L1 ,L2),L3),L4),L5),L6),L7),L8)-L))
2	COND LV(I1,L1) .AND. LV(I2,L2) .AND. LV(I3,L3) .AND. LV(I4,L4) .AND. LV(I5,L5) COND .AND. LV(I6,L6) .AND. LV(I7,L7) .AND. LV(I8,L8) SUBROUTINE CON(I1,I2,I3,I4,I5,I6,I7,I8) CALL CONCAT(I1,I1,I2) CALL CONCAT(I1,I1,I3) CALL CONCAT(I1,I1,I4) CALL CONCAT(I1,I1,I5) .NOT. (I6 .EQ. 0) CALL CONCAT(I1,I1,I6) CALL CONCAT(I1,I1,I7) CALL CONCAT(I1,I1,I8) COND LV(I1,L) .AND. COND L=COMP(COMP(COMP(COMP(COMP(COMP(COMP(L1,L2),L3),L4),L5),L6),L7),L8)	((LV(I1,L1)) .AND. (LV(I2,L2)) .AND. (LV(I3,L3)) .AND. (LV(I4,L4)) .AND. (LV(I5,L5)) .AND. (LV(I6,L6)) .AND. (LV(I7,L7)) .AND. (LV(I8,L8)) .AND. (0 .NE. I6)) .IMPLIES. ((LV(I1,L)) .AND. (0 .EQ. COMP(COMP(COMP(COMP(COMP(COMP(COMP(L1 ,L2),L3),L4),L5),L6),L7),L8)-L))

2.27) CON(I1,I2,I3,I4,I5,I6,I7,I8)

This subroutine has the effect of concatenating the strings I2 to I8 and setting I1 as a pointer to the concatenated string. A call to CON takes the place of 7 different calls to CONCAT. This subroutine can be used to concatenate fewer than 7 strings if desired by setting the extra arguments to 0 to indicate the concatenation of the null strings.

The verification conditions of the subroutine are:

Path 1 This path proves that strings I2 to I5 are concatenated together. It is equivalent to verifying the correctness of subroutine CONCAT(I,J,K) in which CONCAT is called four consecutive times in the path. The result of each call is LI=COMP(LJ,LK) where LI,LJ,LK are list values of lists pointed to by I, J, K respectively.

Path 2 The verification condition of this path is the same as Path 1 except there are seven calls to CONCAT instead of four.

```
      SUBROUTINE SUBST(ITUTE )
CC    LENGTH(L) = (IF NIL THEN 0 ELSE LENGTH(CDR(L))+1)
C     LV(LP,L) MEANS THE LIST VALUE OF LP IS L
CC    LV(LP,L) = (FF(K,(1..M)), ((1.LE.K(I), K(I).LE.5000,
CC    SPACE(2*K(I)-1) = I(*), SPACE(2*K(I)) = 2*K(I+1) -1),
CC    I .IN. (1..M)), LP = 2*K(1) - 1)
      COMMON/BLOCK/SPACE(10000),FAIL,FREE
      INTEGER SPACE,FAIL,FREE,ISYM(14,2),IREP(75)
      DATA ISYM/2HAN,2HOR,2HNO,2HIF,2HLE,2HGE,2HLT,2HGT,2HNE,1H=,2HTR,
     1    2HFA,2HUP,0,1,7,12,18,28,33,38,43,48,53,58,65,73,75    /
      DATA IREP/1H ,1H.,1HD,1HN,1HA,1H.,1H ,1H.,1HR,1HO,1H.,1H ,1H.,1HT,
     1          1HC,1HP,1H.,1H ,1H.,1HS,1HR,1HI,1HL,1HP,1HM,1HI,1H.,1H ,
     2          1H.,1HE,1HL,1H.,1H ,1H.,1HF ,1HG,1H.,1H ,1H.,1HT,1HL,1H.,
     3          1H ,1H.,1HT,1HO,1H.,1H ,1H.,1HT,1HU,1H.,1H ,1H.,1HO,1HF,
     4          1H.,1H ,1H.,1HF,1HU,1HN,1HT,1H.,1H ,1H.,1HF,1HS,1HL,1HA,
     5          1HE,1H.,1H ,1H*,1H*/
CONDSET= (1..10000) .AND. LV(ITUTE,LL) .AND. LL=L0
      K=ITUTE
      MARK=0
4     JDEX = 1
COND  (ELT(MARK+1,L0)=SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,L0)
COND  .NE.ISYM(J,1)), M=1,MARK) .AND. J.NE.10 .AND. J.IN.(1..13))
1     IF(SPACE(K).EQ.ISYM(JDEX,1))GO TO 5
      JDEX = JDEX + 1
      IF ( JDEX.LE.13) GO TO 1
6     K=SPACE(K+1)
      IF (K.EQ.0) GO TO 10
      MARK=MARK+1
      GO TO 4
COND  ELT(MARK+1,L0)=SPACE(K)=ISYM(JDEX,1) .AND. JDEX.IN.(1..13)
5      CALL BAL2(ITUTE,K)
      IF(FAIL.EQ.0.AND.JDEX.EQ.10)GO TO 6
      JDEX1=ISYM(JDEX,2)
      JDEX2=ISYM(JDEX+1,2)
      LDEX = JDEX1
COND  LV(ITUTE,L0) .ST. (SPACE(K)=ELT(MARK+LDEX-JDEX1+1,L0) .AND.
COND  LDEX.IN.(JDEX1..JDEX2))
3     CALL INSERT(ITUTE,MARK,IREP(LDEX))
      LDEX = LDEX + 1
      IF ( LDEX.LE.JDEX2 ) GO TO 3
      KK=SPACE(K+1)
      CALL DELETE(ITUTE,K,K)
      K=KK
      IF (K.EQ.0) GO TO 10
      MARK=MARK+JDEX2-JDEX1+1
       GO TO 4
COND  LV(ITUTE,L) .ST. (((ELT(M,L).NE.ISYM(J,1)), M=1,LENGTH(L))
COND  .AND. J.NE.10 .AND. J.IN.(1..13))
10    RETURN
      END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
SUBROUTINE SUBST(ITUTE) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE,ISYM(14,2),IREP(75) DATA ISYM/2HAB,2HCD,2HND,2HIM,2HLE,2HGE,2HLT,2HGT,2HNE,1H=,2HTR, [DATA ISYM/2HAB,2HCD,2HND,2HIM,2HLE,2HGE,2HLT,2HGT,2HNE,1H=] [ ] 1 2HFA,2HUP,0,1,7,12,18,28,33,38,43,48,53,58,65,73,75 / DATA IREP/1H ,1H(,1HD,1HN,1HA,1H.,1H ,1H.,1HF,1HD,1H.,1H ,1H.,1HT, 1 1HO,1HR,1H.,1H ,1H.,1HS,1HF,1HI,1HL,1HP,1HM,1HI,1H.,1H , 2 1H.,1HE,1HL,1H.,1H ,1H.,1HF,1HG,1H..1H ,1H.,1HT,1HL,1H., 3 1H ,1H.,1HT,1HG,1H.,1H ,1H.,1HE,1HN,1H..1H ,1H.,1HQ,1HF, 4 1H.,1H ,1H.,1HF,1HU,1HS,1HT,1H.,1H ,1H.,1HE,1HS,1HL,1HA, 5 1HF,1H.,1H ,1H*,1HW/		
COND SET=(1..10000) .AND. LV(ITUTE,L) .AND. LL=LO K=ITUTE MARK=0 4 JDEX=1 COND (FLT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((FLT(M,LO) COND .NE. ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))	[K] [MARK] [JDEX]	[ITUTE] [ ] [ ]

1 IF(SPACE(K) .EQ. ISYM(JDEX,1))GO TO 5
JDEX=JDEX+1
IF(JDEX .LE. 13)GO TO 1
4 K=SPACE(K+1)
IF(K .EQ. 0)GO TO 10
MARK=MARK+1
GO TO 4
	[JDEX]
	[K]
	[MARK]
		[JDEX]
		[SPACE,K]
		[MARK]

COND FLT(MARK+1,LO)=SPACE(K)=ISYM(JDEX,1) .AND. JDEX.IN.(1..13) 5 CALL FAIL2(ITUTE,K) IF(FAIL .EQ. 0 .AND. JDEX .EQ. 10)GO TO 6 JDEX1=ISYM(JDEX,2) JDEX2=ISYM(JDEX+1,2)	[JDEX1] [JDEX2]	[ISYM,JDEX] [ISYM,JDEX]
LDEX=JDEX1 COND LV(ITUTE,LO).ST.(SPACE(K) .EQ. FLT(MARK+LDEX-JDEX+1,LO) .AND. COND LDEX.IN.(JDEX1..JDEX2)) 3 CALL INSERT(ITUTE,MARK,IREP(LDEX)) LDEX=LDEX+1 IF(LDEX .LE. JDEX2)GO TO 3	[LDEX]	[LDEX]
KK=SPACE(K+1) CALL DELETE(ITUTE,K,K) K=KK IF(K .EQ. 0)GO TO 10 MARK=MARK+JDEX2-JDEX1+1 GO TO 4	[KK] [K] [MARK]	[SPACE,K] [KK] [MARK,JDEX2,JDEX1]
COND LV(ITUTE,L).ST.(((FLT(M,L) .NE. ISYM(J,1)),M .EQ. 1,LENGTH(L)) COND .AND. J .NE. 10 .AND. J.IN.(1..13)) 10 RETURN END		

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1		
	COND .TRUE.	
	SUBROUTINE SUBST(ITUTE)	
	COMMON/BLOCK/SPACE(10000),FAIL,FREE	
	INTEGER SPACE,FAIL,FREE,ISYM(14,2),IREP(75)	
	DATA ISYM/2HAM,2HOP,2HNO,2HIM,2HLE,2HGE,2HLT,2HGT,2HNE,1H=,2HTR,	
	1 2HFA,2HUP,0,1,7,12,18,28,33,38,43,48,53,58,65,73,75 /	
	DATA IREP/1H ,1H.,1HO,1HN,1HA,1H.,1H ,1H.,1HF,1HO,1H.,1H ,1H.,1HT,	
	1 1HC,1HN,1H.,1H ,1H.,1HS,1HE,1HT,1HL,1HP,1HM,1HI,1H.,1H ,	
	2 1H.,1HF,1HL,1H.,1H ,1H.,1HS,1HG,1H.,1H ,1H.,1HT,1HL,1H.,	
	3 1H ,1H.,1HT,1HG,1H.,1H ,1H.,1HF,1HN,1H.,1H ,1H.,1HO,1HE,	
	4 1H.,1H ,1H.,1HE,1HU,1HR,1HT,1H.,1H ,1H.,1HE,1HS,1HL,1HA,	
	5 1HE,1H.,1H ,1H*,1H*/	
	COND SET=(1..10000) .AND. LV(ITUTE,LL) .AND. LL=LO	
		(0 .EQ. 1..10000-SET) .AND. (LV(ITUTE,LL)) .AND. (0 .EQ. LO-LL)
2		
	COND SET=(1..10000) .AND. LV(ITUTE,LL) .AND. LL=LO	
	K=ITUTE	
	MARK=0	
4	JDEX=1	
	COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE. COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))	
		((0 .EQ. 1..10000-SET) .AND. (LV(ITUTE,LL)) .AND. (0 .EQ. LO-LL) .AND. (((0 .EQ. -ELT(1,LO)+SPACE(ITUTE))) .IMPLIES. ((ITJTE)))) .IMPLIES. (((J.IN.) .AND. (1..13)) .AND. (0 .EQ. ELT(M,LO)-ISYM(J,1),M+1,0))
3		
	COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE. COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))	
	1 (SPACE(K) .EQ. ISYM(JDEX,1))	
2	COND ELT(MARK+1,LO)=SPACE(K)=ISYM(JDEX,1) .AND. JDEX.IN.(1..13)	
		((0 .EQ. -ELT(1+MARK,LO)+SPACE(K)) .AND. (0 .EQ. ISYM(JDEX,1)-SPACE(K))) .IMPLIES. ((JDEX.IN.) .AND. (1..13))
2		
4		
	COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE. COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))	
	1 .AND. (SPACE(K) .EQ. ISYM(JDEX,1))	
1	JDEX=JDEX+1	
	(JDEX .LE. 13)	

```
COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE.
COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))
                                                        ((0 .EQ. -ELT(1+MARK,LO)+SPACE(K))
                                                        .AND. (K.IN.SET)) .IMPLIES.
                                                        (((J.IN.) .AND. (1..13)) .AND. (0
                                                        .EQ. ELT(M,LO)-ISYM(J,1),M+1,MARK))

5
      COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE.
      COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))
      1       .NOT. (SPACE(K) .EQ. ISYM(JDEX,1))
              JDEX=JDEX+1
               .NOT. (JDEX .LE. 13)
      6       K=SPACE(K+1)
              (K .EQ. 0)
      COND (V(ITUTE,L).ST.(((ELT(M,L) .NE. ISYM(J,1)),M .EQ. 1,LENGTH(L)) .AND. J
      COND .NE. 10 .AND. J.IN.(1..13))
                                                        ((0 .EQ.
                                                        ELT(M,L)-ISYM(J,1),M+1,LENGTH(L))
                                                        .AND. (0 .NE. 10-J)) .IMPLIES.
                                                        ((J.IN.) .AND. (1..13))

6
      COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE.
      COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))
      1       .NOT. (SPACE(K) .EQ. ISYM(JDEX,1))
              JDEX=JDEX+1
               .NOT. (JDEX .LE. 13)
      6       K=SPACE(K+1)
               .NOT. (K .EQ. 0)
              MARK=MARK+1
      4       JDEX=1
      COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE.
      COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))
                                                        ((0 .EQ.
                                                        ELT(M,LO)-ISYM(J,1),M+1,MARK) .AND.
                                                        (0 .NE. 10-J) .AND. ((J.IN.) .AND.
                                                        (1..13)) .AND. ( .NOT. (0 .EQ.
                                                        ISYM(JDEX,1)-SPACE(K))) .AND. (
                                                        .NOT. (0 .LE. 12-JDEX)) .AND. (
                                                        .NOT. (0 .EQ. SPACE(1+K))) .AND.
                                                        (((0 .EQ.
                                                        -ELT(2+MARK,LO)+SPACE(SPACE(1+K))))
                                                        .IMPLIES. ((SPACE(1+K))))) .IMPLIES.
                                                        (((J.IN.) .AND. (1..13)) .AND. (0
                                                        .EQ. ELT(M,LO)-ISYM(J,1),M+1,1+MARK))

7
      COND ELT(MARK+1,LO)=SPACE(K)=ISYM(JDEX,1) .AND. JDEX.IN.(1..13)
      5       CALL BAL2(ITUTE,K)
              (FAIL .EQ. 0 .AND. JDEX .EQ. 10)
      6       K=SPACE(K+1)
              (K .EQ. 0)
```

```
      COND LV(ITUTE,L).ST.(((ELT(M,L) .NE. ISYM(J,1)),M .EQ. 1,LENGTH(L)) .AND. J
      COND .NE. 10 .AND. J.IN.(1..13))
                                                                    ((0 .EQ.
                                                                    ELT(M,L)-ISYM(J,1),M+1,LENGTH(L))
                                                                    .AND. (0 .NE. 10-J)) .IMPLIES.
                                                                    ((J.IN.) .AND. (1..13))

  8
      COND ELT(MARK+1,LO)=SPACE(K)=ISYM(JDEX,1) .AND. JDEX.IN.(1..13)
      5     CALL BAL?(ITUTE,K)
            (FAIL .EQ. 0 .AND. JDEX .EQ. 10)
      6     K=SPACE(K+1)
            .NOT. (K .EQ. 0)
            MARK=MARK+1
      4     JDEX=1
      COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE.
      COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))
                                                                    ((JDEX.IN.) .AND. (1..13) .AND. ((0
                                                                    .EQ. FAIL) .AND. (0 .EQ. 10-JDEX))
                                                                    .AND. ( .NOT. (0 .EQ. SPACE(1+K)))
                                                                    .AND. (((0 .EQ.
                                                                    -ELT(2+MARK,LO)+SPACE(SPACE(1+K))))
                                                                    .IMPLIES. ((SPACE(1+K))))) .IMPLIES.
                                                                    (((J.IN.) .AND. (1..13)) .AND. (0
                                                                    .NE. 10-J)) .AND. (0 .EQ.
                                                                    ELT(M,LO)-ISYM(J,1),M+1,1+MARK))

  9
      COND ELT(MARK+1,LO)=SPACE(K)=ISYM(JDEX,1) .AND. JDEX.IN.(1..13)
      5     CALL BAL?(ITUTE,K)
            .NOT. (FAIL .EQ. 0 .AND. JDEX .EQ. 10)
            JDEX1=ISYM(JDEX,2)
            JDEX2=ISYM(JDEX+1,2)
            LDEX=JDEX1
      COND LV(ITUTE,LO).ST.(SPACE(K) .EQ. ELT(MARK+LDEX-JDEX+1,LO) .AND.
      COND LDEX.IN.(JDEX1..JDEX2))
                                                                    ((0 .EQ. -ELT(1+MARK,LO)+SPACE(K))
                                                                    .AND. (0 .EQ.
                                                                    ISYM(JDEX,1)-SPACE(K))) .IMPLIES.
                                                                    ((((((0 .EQ.
                                                                    ELT(1+ISYM(JDEX,2)-JDEX+MARK,LO)-SPAC
                                                                    E(K))) .IMPLIES. (((ISYM(JDEX,2))))
                                                                    .IMPLIES. (((ISYM(JDEX,2))))) .AND.
                                                                    (((JDEX.IN.) .AND. (1..13) .AND. (
                                                                    .NOT. ((0 .EQ. FAIL) .AND. (0 .EQ.
                                                                    10-JDEX)))) .IMPLIES.
                                                                    ((LV(ITUTE,LO)))))

 10
      COND LV(ITUTE,LO).ST.(SPACE(K) .EQ. ELT(MARK+LDEX-JDEX+1,LO) .AND.
      COND LDEX.IN.(JDEX1..JDEX2))
      3     CALL INSERT(ITUTE,MARK,IREP(LDEX))
            LDEX=LDEX+1
```

```
      (LDEX .LE. JDEX2)
COND LV(ITUTE,LO).ST.(SPACE(K) .EQ. ELT(MARK+LDEX-JDEX+1,LO) .AND.
COND LDEX.IN.(JDEX1..JDEX2))
 ***** INPUT STRING IS NOT A NUMBER *****
```

```
(((LDEX.IN.) .AND. (JDEX1..JDEX2)
.AND. (0 .LE. -1+JDEX2)) .IMPLIES.
((LV(ITUTE,LO)))) .IMPLIES.
((JDEX1..JDEX2) .AND. (((0 .EQ.
ELT(2+LDEX-JDEX+MARK,LO)-SPACE(K)))
.IMPLIES. ((1+LDEX))))
```

```
11
      (
COND LV(ITUTE,LO).ST.(SPACE(K) .EQ. ELT(MARK+LDEX-JDEX+1,LO) .AND.
COND (DEX.IN.(JDEX1..JDEX2))
3     CALL INSERT(ITUTE,MARK,IREP(LDEX))
      LDEX=LDEX+1
       .NOT. (LDEX .LE. JDEX2)
      KK=SPACE(K+1)
      CALL DELETE(ITUTE,K,K)
      K=KK
      (K .EQ. 0)
COND LV(ITUTE,L).ST.(((ELT(M,L) .NE. ISYM(J,1)),M .EQ. 1,LENGTH(L)) .AND. J
COND .NE. 10 .AND. J.IN.(1..13))
 ***** INPUT STRING IS NOT A NUMBER *****
 ***** INPUT STRING IS NOT A NUMBER *****
```

```
((0 .EQ. 1,LENGTH(L)-0 .NE.
ISYM(J,1),M) .AND. (0 .NE. 10-J))
.IMPLIES. ((J.IN.) .AND. (1..13))
```

```
12
COND LV(ITUTE,LO).ST.(SPACE(K) .EQ. ELT(MARK+LDEX-JDEX+1,LO) .AND.
COND (DEX.IN.(JDEX1..JDEX2))
3     CALL INSERT(ITUTE,MARK,IREP(LDEX))
      LDEX=LDEX+1
       .NOT. (LDEX .LE. JDEX2)
      KK=SPACE(K+1)
      CALL DELETE(ITUTE,K,K)
      K=KK
       .NOT. (K .EQ. 0)
      MARK=MARK+JDEX2-JDEX1+1
4     JDEX=1
3 COND (ELT(MARK+1,LO) .EQ. SPACE(K) .AND. K.IN.SET) .AND. (((ELT(M,LO) .NE.
COND ISYM(J,1)),M .EQ. 1,MARK) .AND. J .NE. 10 .AND. J.IN.(1..13))
 ***** INPUT STRING IS NOT A NUMBER *****
 ***** INPUT STRING IS NOT A NUMBER *****
 ***** INPUT STRING IS NOT A NUMBER *****
 ***** INPUT STRING IS NOT A NUMBER *****
```

```
2
((LDEX.IN.) .AND. (JDEX1..JDEX2)
.AND. ( .NOT. (0 .LE. -1+JDEX2))
.AND. ( .NOT. (0 .EQ. SPACE(1+K)))
.AND. (((0 .EQ. SPACE(K))) .IMPLIES.
((SPACE(1+K))))) .IMPLIES. (((J.IN.)
.AND. (1..13)) .AND. (0 .NE. 10-J)
1 .AND. (0 .EQ. 1,MARK=0 .NE.
```

ISYM(J,I),M))

2.28) SUBST(ITUTE)

ITUTE is a string with tokens in it representing certain boolean and relational operators. SUBST replaces these tokens by appropriate strings to make the expression easier to read.

".AND."	replaces	"∧"
".OR."	"	"∨"
".NOT."	"	"¬"
".IMPLIES."	"	"→"
".LE."	"	"≤"
".GE."	"	"≥"
".LT."	"	"<"
".GT."	"	">"
".NE."	"	"≠"
".EQ."	"	"="
".TRUE."	"	"T"
".FALSE."	"	"F"
"**"	"	"↑"

The assertions written for these paths are slightly insufficient. Some refinement must be made for the program to be verifiable.

```
CC      PARLIST(LP1,[illegible],L1)=SUBLIST(LP1,ELT(I,L1),L1) .ST.
CC      COM.(I-1,L1)=LV(LP2,L2)
C[illegible]      LV(ICOMP,L1),LV(ICOMP,L2)
        SU[illegible]                (ICOMP1,ICOMP2)
        C[illegible]                (1000),FAIL,IFOO
        [illegible]                ,ICOMP2,C1,C2,SPACE,FAIL
        CALL SLIST(C1,0)
        CALL SLIST(C2,0)
        CALL COPY(C[illegible],ICOMP1,0)
        CALL COPY(C[illegible],ICOMP2,0)
C[illegible]      K=0,PARLIST(ICOMP,C[illegible],L1),PARLIST(ICOMP2,COMP2,L2)
100     IF([illegible]
        SPACE(C1)=SPACE(COMP1)
        CALL DELETE(COMP1,0)
        IF(COMP2.EQ.0) GO TO 600
        SPACE(C2)=SPACE(COMP2)
        CALL COPY(COMP2,0)
        CALL PUT(C1,C2,0)
        IF(F)[illegible]
900     FAIL=0
        GO TO 700
800     FAIL=1
700     CALL ERASE(C1)
        CALL ERASE(C2)
        CALL ERASE(COMP)
        CALL DELETE(COMP2)
C[illegible]      FAIL=0 .OR. FAIL=1
        RETURN
        END
```

		EFFECTIVE RANGE	EFFECTIVE DOMAIN
		[SPACE]	[SPACE,C1,COMP1]
		[SPACE]	[SPACE,C2,COMP2]
		[FAIL]	[]
		[FAIL]	[]

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	(CMP.LV(ICOMP,L1) .AND. LV(ICOMP,L2) SUB ENTRY ACCMP(ICOMP1,ICOMP2) COMMON/BLOCK/SPACE(10000),FAIL,IFOO DIMENSION SPACE,COMP1,C2,SPACE,FAIL CALL PLIST(C1,0) CALL PLIST(C2,0) CALL COPY(COMP1,ICOMP1,0) CALL COPY(COMP2,ICOMP2,0) COMP K=0 .AND. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2) 1	((LV(ICOMP,L1)) .AND. (LV(ICOMP,L2))) .IMPLIES. ((0 .EQ. K) .AND. (PARLIST(ICOMP,COMP1,L1)) .AND. (PARLIST(ICOMP2,COMP2,L2)))
2	COMP K=0 .AND. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2) 100 (COMP1 .EQ. 0) 500 FAIL=0 700 CALL PLIST(C1) CALL PLIST(C2) CALL PLIST(COMP1) CALL PLIST(COMP2) C=0 FAIL=0 .OR. FAIL=1	NONE
3	COMP K=0 .AND. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2) 100 .NOT. (ICOMP1 .EQ. 0) SPACE(C1)=SPACE(COMP1) CALL PLIST(COMP1,0) (COMP2 .EQ. 0) 500 FAIL=1 700 CALL PLIST(C1) CALL PLIST(C2) CALL PLIST(COMP1) CALL PLIST(COMP2) C=0 FAIL=0 .OR. FAIL=1	NONE
4	COMP K=0 .AND. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2) 100 .NOT. (ICOMP1 .EQ. 0) SPACE(C1)=SPACE(COMP1) CALL PLIST(COMP1,0) .NOT. (COMP2 .EQ. 0) SPACE(C2)=SPACE(COMP2) CALL PLIST(COMP2,0)	

```
      IF (C1,C2,K)
     (K .LT. 0)
600  FAIL=1
700  CALL PARLIST(C1)
     CALL L1(P2)
     CALL PARLIST(COMP1)
     CALL L2(COMP2)
     IF FAIL=0 .OR. FAIL=1
```

NONE

5

```
     IF K=0 .AND. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2)
100    . T. (COMP1 .EQ. 0)
       C1=SPACE(COMP1)
       CALL DELETE(COMP1,0)
       . T. (COMP2 .EQ. 0)
       C2=SPACE(COMP2)
       CALL DELETE(COMP2,0)
       CALL PUT(C1,C2,K)
       . T. (K .LT. 0)
       (K .EQ. 0)
       K=0 .OR. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2)
```

((0 .EQ. K) .AND. (PARLIST(ICOMP,COMP1,L1)) .AND. (PARLIST(ICOMP2,COMP2,L2)) .AND. (0 .NE. COMP1) .AND. (0 .NE. COMP2) .AND. (0 .LE. K)) .IMPLIES. ((PARLIST(ICOMP,COMP1,L1)) .AND. (PARLIST(ICOMP2,COMP2,L2)))

5

```
     K=0 .OR. PARLIST(ICOMP,COMP1,L1) .AND. PARLIST(ICOMP2,COMP2,L2)
100    . T. (COMP1 .EQ. 0)
       C1=SPACE(COMP1)
       CALL DELETE(COMP1,0)
       . T. (COMP2 .EQ. 0)
       C2=SPACE(COMP2)
       CALL PUT(C1,C2,K)
       . T. (K .LT. 0)
       . T. (K .EQ. 0)
600    FAIL=0
700    CALL PARLIST(C1)
       CALL L1(C2)
       CALL PARLIST(COMP1)
       CALL L2(COMP2)
       IF FAIL=1 .OR. FAIL=1
```

NONE

BCOMP(ICOMP1,ICOMP2)

The arguments to this subroutine are two variables. FAIL is set to 0 if ICOMP1 precedes or equals ICOMP2 alphabetically and FAIL is set to 1 otherwise.

VERIFICATION:

Path 1: insufficient definition for assertion.

Path 2: proved.

Path 3: proved.

Path 4: proved.

Path 5: insufficient definition for assertion.

Path 6: proved.

```
COND    LV(IREL,I1),LV(LISTL,L2),LISTL .NE. 0
        SUBROUTINE SEEK(IREL,LISTL)
        COMMON/BLOCK/SPACE(10000),FAIL,FREE
        INTEGER SPACE,FAIL,FREE
        DATA LE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/
        CALL DIFF(IREL,0,0,0)
        LIS=LISTL
        CALL LEN(IREL,3,K)
        IF(SPACE(K).EQ.LT)GO TO 10
        IF(SPACE(K).EQ.LE)GO TO 20
        IF(SPACE(K).EQ.IEQ)GO T O 30
        IF(SPACE(K).EQ.NE)GO TO 40
100     FAIL=1
COND     (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,
COND     SPACE(K) .NE. NE) .OR. (LIS .LE. 0)
        RETURN
COND     LIS .NE. 0,SPACE(K)=LT
10      LOOK=SPACE(LIS)
        CALL DIFF(IREL,LOOK,ISIGN,IOP)
        IF(ISIGN.EQ.1 .AND. IOP .NE. NE)GO TO 200
        IF(ISIGN.EQ.0 .AND.IOP .EQ. LT) GO TO 200
        LIS=SPACE(LIS+1)
        IF(LIS)100,100,10
COND     LIS .NE. 0,SPACE(K)=LE
20      LOOK = SPACE(LIS)
        CALL DIFF(IREL,LOOK,ISIGN,IOP)
        IF((ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE. NE)GO TO 200
        IF(IOP.EQ.LT.AND.ISIGN.EQ.-1) GO TO 200
        LIS=SPACE(LIS+1)
        IF(LIS)100,100,20
COND     LIS .NE. 0,SPACE(K)=IEQ
30      LOOK = SPACE(LIS)
        CALL DIFF(IREL,LOOK,ISIGN,IOP)
        IF(ISIGN.EQ.0 .AND. IOP .EQ. IEQ)GO TO 200
        LIS = SPACE(LIS+1)
        IF(LIS)100,100,30
COND     LIS .NE. 0,SPACE(K)=NE
40      LOOK=SPACE(LIS)
        CALL DIFF(IREL,LOOK,ISIGN,IOP)
        IF(IOP.EQ.IEQ.AND.(ISIGN.EQ.-2.OR.ISIGN.EQ.1))GO TO 200
        LIS=SPACE(LIS+1)
        IF(LIS)100,100,40
200     FAIL=0
COND     SPACE(K)=LT,ISIGN=1,IOP .NE. NE,FAIL=0
        RETURN
  201   FAIL=0
COND     SPACE(K)=LT,ISIGN=0,IOP=LT,FAIL=0
        RETURN
  202   FAIL=0
COND     SPACE(K)=LE,(ISIGN=0 .OR. ISIGN=1),IOP .NE. NE,FAIL=0
        RETURN
  203   FAIL=0
COND     SPACE(K)=LE,ISIGN=-1,IOP=LT,FAIL=0
        RETURN
  204   FAIL=0
```

```
COND    SPACE(K)=IEQ,ISIGN=0,IOP=IEQ,FAIL=0
        RETURN
  205   FAIL=0
COND     SPACE(K)=NE,(ISIGN=-2 .OR. ISIGN=1),IOP=IEQ,FAIL=0
        RETURN
        END
```

PROGRAM TO BE VERIFIED	EFFECTIVE RANGE	EFFECTIVE DOMAIN
COND LV(IREL,I1) .AND. LV(LISTL,I2) .AND. LISTL .NE. 0		
SUBROUTINE SEEK(IREL,LISTL)		
COMMON/BLOCK/SPACE(10000),FAIL,FREE		
INTEGER SPACE,FAIL,FREE		
DATALE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/	[DATALE/2HLE/,LT/2HLT/,IEQ/1H] [NE]	
CALL DIFF(IREL,0,0,0)		
LIS=LISTL	[LIS]	[LISTL]
CALL LEN(IREL,3,K)		
IF(SPACE(K) .EQ. LT)GO TO 10		
IF(SPACE(K) .EQ. LE)GO TO 20		
IF(SPACE(K) .EQ. IEQ)GO TO 30		
IF(SPACE(K) .EQ. NE)GO TO 40		
100 FAIL=1	[FAIL]	[]
COND (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,SPACE(K) .NE.		
COND NE) .OR. (LIS .LE. 0)		
RETURN		
COND LIS .NE. 0 .AND. SPACE(K)=LT		
10 LOOK=SPACE(LIS)	[LOOK]	[SPACE,LIS]
CALL DIFF(IREL,LOOK,ISIGN,IOP)		
IF(ISIGN .EQ. 1 .AND. IOP .NE. NE)GO TO 200		
IF(ISIGN .EQ. 0 .AND. IOP .EQ. LT)GO TO 200		
LIS=SPACE(LIS+1)	[LIS]	[SPACE,LIS]
IF(LIS .LE. 0)GO TO 100		
GO TO 10		
COND LIS .NE. 0 .AND. SPACE(K)=LE		
20 LOOK=SPACE(LIS)	[LOOK]	[SPACE,LIS]
CALL DIFF(IREL,LOOK,ISIGN,IOP)		
IF((ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE. NE)GO TO 200		
IF(IOP.EQ .LT. AND.ISIGN .EQ. -1)GO TO 200		
LIS=SPACE(LIS+1)	[LIS]	[SPACE,LIS]
IF(LIS .LE. 0)GO TO 100		
GO TO 20		
COND LIS .NE. 0 .AND. SPACE(K)=IEQ		
30 LOOK=SPACE(LIS)	[LOOK]	[SPACE,LIS]
CALL DIFF(IREL,LOOK,ISIGN,IOP)		
IF(ISIGN .EQ. 0 .AND. IOP .EQ. IEQ)GO TO 200		
LIS=SPACE(LIS+1)	[LIS]	[SPACE,LIS]
IF(LIS .LE. 0)GO TO 100		
GO TO 30		
COND LIS .NE. 0 .AND. SPACE(K)=NE		
40 LOOK=SPACE(LIS)	[LOOK]	[SPACE,LIS]
CALL DIFF(IREL,LOOK,ISIGN,IOP)		
IF(IOP .EQ. IEQ .AND. (ISIGN .EQ. -2 .OR. ISIGN .EQ. 1))GO TO 200		
LIS=SPACE(LIS+1)	[LIS]	[SPACE,LIS]
IF(LIS .LE. 0)GO TO 100		
GO TO 40		
200 FAIL=0	[FAIL]	[]
COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0		
RETURN		

```
201   FAIL=0                                                    [FAIL]          []
COND SPACE(K)=LT .AND. ISIGN=0 .AND. IOP=LT .AND. FAIL=0
      RETURN
202   FAIL=0                                                    [FAIL]          []
COND SPACE(K)=LE .AND. (ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE.
COND NE .AND. FAIL=0
      RETURN
203   FAIL=0                                                    [FAIL]          []
COND SPACE(K)=LE .AND. ISIGN=-1 .AND. IOP=LT .AND. FAIL=0
      RETURN
204   FAIL=0                                                    [FAIL]          []
COND SPACE(K)=IEQ .AND. ISIGN=0 .AND. IOP=IEQ .AND. FAIL=0
      RETURN
205   FAIL=0                                                    [FAIL]          []
COND SPACE(K)=NE .AND. (ISIGN .EQ. -2 .OR. ISIGN .EQ. 1) .AND. IOP=IEQ
COND .AND. FAIL=0
      RETURN
      END
```

PATH NUMBER	PATH	STATEMENTS REMAINING TO BE VERIFIED
1	COND LV(IREL,I1) .AND. LV(LISTL,L2) .AND. LISTL .NE. 0 SUBROUTINE SEEK(IREL,LISTL) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE DATA LE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/ CALL DIFF(IREL,0,0,0) LIS=LISTL CALL LEN(IREL,3,K) (SPACE(K) .EQ. LT) COND LIS .NE. 0 .AND. SPACE(K)=LT	((LV(IREL,I1)) .AND. (LV(LISTL,L2)) .AND. (0 .NE. -LISTL) .AND. (0 .EQ. LT-SPACE(K))) .IMPLIES. ((0 .NE. -LISTL))
2	COND LV(IREL,I1) .AND. LV(LISTL,L2) .AND. LISTL .NE. 0 SUBROUTINE SEEK(IREL,LISTL) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE DATA LE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/ CALL DIFF(IREL,0,0,0) LIS=LISTL CALL LEN(IREL,3,K) .NOT. (SPACE(K) .EQ. LT) (SPACE(K) .EQ. LE) COND LIS .NE. 0 .AND. SPACE(K)=LE	((LV(IREL,I1)) .AND. (LV(LISTL,L2)) .AND. (0 .NE. -LISTL) .AND. (0 .NE. LT-SPACE(K)) .AND. (0 .EQ. LE-SPACE(K))) .IMPLIES. ((0 .NE. -LISTL))
3	COND LV(IREL,I1) .AND. LV(LISTL,L2) .AND. LISTL .NE. 0 SUBROUTINE SEEK(IREL,LISTL) COMMON/BLOCK/SPACE(10000),FAIL,FREE INTEGER SPACE,FAIL,FREE DATA LE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/ CALL DIFF(IREL,0,0,0) LIS=LISTL CALL LEN(IREL,3,K) .NOT. (SPACE(K) .EQ. LT) .NOT. (SPACE(K) .EQ. LE) (SPACE(K) .EQ. IEQ) COND LIS .NE. 0 .AND. SPACE(K)=IEQ	

```
                                                                      ((LV(IREL,I1)) .AND. (LV(LISTL,L2))
                                                                      .AND. (0 .NE. -LISTL) .AND. (0 .NE.
                                                                      LT-SPACE(K)) .AND. (0 .NE.
                                                                      LE-SPACE(K)) .AND. (0 .EQ.
                                                                      IEQ-SPACE(K)))
                                                                       .IMPLIES.
                                                                      ((0 .NE. -LISTL))

4
      COND LV(IREL,I1) .AND. LV(LISTL,L2) .AND. LISTL .NE. 0
            SUBROUTINE SEEK(IREL,LISTL)
            COMMON/BLOCK/SPACE(10000),FAIL,FREE
            INTEGER SPACE,FAIL,FREE
            DATALE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/
            CALL DIFF(IREL,0,0,0)
            LIS=LISTL
            CALL LEN(LISTL,3,K)
             .NOT. (SPACE(K) .EQ. LT)
             .NOT. (SPACE(K) .EQ. LE)
             .NOT. (SPACE(K) .EQ. IEQ)
            (SPACE(K) .EQ. NE)
      COND LIS .NE. 0 .AND. SPACE(K)=NE
                                                                      ((LV(IREL,I1)) .AND. (LV(LISTL,L2))
                                                                      .AND. (0 .NE. -LISTL) .AND. (0 .NE.
                                                                      LT-SPACE(K)) .AND. (0 .NE.
                                                                      LE-SPACE(K)) .AND. (0 .NE.
                                                                      IEQ-SPACE(K)) .AND. (0 .EQ.
                                                                      NE-SPACE(K)))
                                                                       .IMPLIES.
                                                                      ((0 .NE. -LISTL))

5
      COND LV(IREL,I1) .AND. LV(LISTL,L2) .AND. LISTL .NE. 0
            SUBROUTINE SEEK(IREL,LISTL)
            COMMON/BLOCK/SPACE(10000),FAIL,FREE
            INTEGER SPACE,FAIL,FREE
            DATALE/2HLE/,LT/2HLT/,IEQ/1H=/,NE/2HNE/
            CALL DIFF(IREL,0,0,0)
            LIS=LISTL
            CALL LEN(IREL,3,K)
             .NOT. (SPACE(K) .EQ. LT)
             .NOT. (SPACE(K) .EQ. LE)
             .NOT. (SPACE(K) .EQ. IEQ)
             .NOT. (SPACE(K) .EQ. NE)
      100   FAIL=1
      COND (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,SPACE(K) .NE. NE)
      COND .OR. (LIS .LE. 0)
                                                                      ((LV(IREL,I1)) .AND. (LV(LISTL,L2))
                                                                      .AND. (0 .NE. -LISTL) .AND. (0 .NE.
                                                                      LT-SPACE(K)) .AND. (0 .NE.
                                                                      LE-SPACE(K)) .AND. (0 .NE.
                                                                      IEQ-SPACE(K)) .AND. (0 .NE.
                                                                      NE-SPACE(K)))
                                                                       .IMPLIES.
```

```
                                                                    ((0 .NE. NE+SPACE(K)) .OR. (0 .LE.
                                                                    -LISTL))

6
      COND LIS .NE. 0 .AND. SPACE(K)=LT
      10    LOOK=SPACE(LIS)
            CALL DIFF(IREL,LOOK,ISIGN,IOP)
            (ISIGN .EQ. 1 .AND. IOP .NE. NE)
      200   FAIL=0
      COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0
                                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                                    LT-SPACE(K)) .AND. ((0 .EQ. 1-ISIGN)
                                                                    .AND. (0 .NE. -IOP+NE)))
                                                                     .IMPLIES.
                                                                    ((0 .NE. -IOP+NE) .AND. (0 .EQ.
                                                                    1-ISIGN))

7
      COND LIS .NE. 0 .AND. SPACE(K)=LT
      10    LOOK=SPACE(LIS)
            CALL DIFF(IREL,LOOK,ISIGN,IOP)
             .NOT. (ISIGN .EQ. 1 .AND. IOP .NE. NE)
            (ISIGN .EQ. 0 .AND. IOP .EQ. LT)
      200   FAIL=0
      COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0
                                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                                    LT-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                                    1-ISIGN) .AND. (0 .NE. -IOP+NE)))
                                                                    .AND. ((0 .EQ. ISIGN) .AND. (0 .EQ.
                                                                    -IOP+LT)))
                                                                     .IMPLIES.
                                                                    ((0 .NE. -IOP+NE) .AND. (0 .EQ.
                                                                    1-ISIGN))

8
      COND LIS .NE. 0 .AND. SPACE(K)=LT
      10    LOOK=SPACE(LIS)
            CALL DIFF(IREL,LOOK,ISIGN,IOP)
             .NOT. (ISIGN .EQ. 1 .AND. IOP .NE. NE)
             .NOT. (ISIGN .EQ. 0 .AND. IOP .EQ. LT)
            LIS=SPACE(LIS+1)
            (LIS .LE. 0)
      100   FAIL=1
      COND (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,SPACE(K) .NE. NE)
      CONE .OR. (LIS .LE. 0)
                                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                                    LT-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                                    1-ISIGN) .AND. (0 .NE. -IOP+NE)))
                                                                    .AND. ( .NOT. ((0 .EQ. ISIGN) .AND.
                                                                    (0 .EQ. -IOP+LT))) .AND. (0 .LE.
                                                                    -SPACE(1+LIS)))
                                                                     .IMPLIES.
                                                                    ((0 .NE. NE+SPACE(K)) .OR. (0 .LE.
```

```
                                                                -SPACE(1+LIS)))

9
      COND LIS .NE. 0 .AND. SPACE(K)=LT
      10    LOOK=SPACE(LIS)
            CALL DIFF(IREL,LOOK,ISIGN,IOP)
             .NOT. (ISIGN .EQ. 1 .AND. IOP .NE. NE)
             .NOT. (ISIGN .EQ. 0 .AND. IOP .EQ. LT)
            LIS=SPACE(LIS+1)
             .NOT. (LIS .LE. 0)
      COND LIS .NE. 0 .AND. SPACE(K)=LT
                                                                ((0 .NE. -LIS) .AND. (0 .EQ.
                                                                LT-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                                1-ISIGN) .AND. (0 .NE. -IOP+NE)))
                                                                .AND. ( .NOT. ((0 .EQ. ISIGN) .AND.
                                                                (0 .EQ. -IOP+LT))) .AND. (0 .LT.
                                                                SPACE(1-LIS)))
                                                                 .IMPLIES.
                                                                ((0 .NE. -SPACE(1+LIS)))

10
      COND LIS .NE. 0 .AND. SPACE(K)=LE
      20    LOOK=SPACE(LIS)
            CALL DIFF(IREL,LOOK,ISIGN,IOP)
            ((ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE. NE)
      200   FAIL=0
      COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0
                                                                ((0 .NE. -LIS) .AND. (0 .EQ.
                                                                LE-SPACE(K)) .AND. (((0 .EQ. ISIGN)
                                                                .OR. (0 .EQ. 1-ISIGN)) .AND. (0 .NE.
                                                                -IOP+NE)))
                                                                 .IMPLIES.
                                                                ((0 .NE. -IOP+NE) .AND. (0 .EQ.
                                                                1-ISIGN) .AND. (0 .EQ. LT-SPACE(K)))

11
      COND LIS .NE. 0 .AND. SPACE(K)=LE
      20    LOOK=SPACE(LIS)
            CALL DIFF(IREL,LOOK,ISIGN,IOP)
             .NOT. ((ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE. NE)
            (IOP.EQ .LT. AND.ISIGN .EQ. -1)
      200   FAIL=0
      COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0
                                                                ((0 .NE. -LIS) .AND. (0 .EQ.
                                                                LE-SPACE(K)) .AND. ( .NOT. (((0 .EQ.
                                                                ISIGN) .OR. (0 .EQ. 1-ISIGN)) .AND.
                                                                (0 .NE. -IOP+NE))) .AND. (0 .EQ.
                                                                -1+IOP.EQ))
                                                                 .IMPLIES.
                                                                ((0 .NE. -IOP+NE) .AND. (0 .EQ.
                                                                1-ISIGN) .AND. (0 .EQ. LT-SPACE(K)))
```

12

```
COND LIS .NE. 0 .AND. SPACE(K)=LE
20    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
       .NOT. ((ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE. NE)
       .NOT. (IOP.EQ .LT. AND.ISIGN .EQ. -1)
      LIS=SPACE(LIS+1)
      (LIS .LE. 0)
100   FAIL=1
COND (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,SPACE(K) .NE. NE)
COND .OR. (LIS .LE. 0)
```

```
((0 .NE. -LIS) .AND. (0 .EQ.
LE-SPACE(K)) .AND. ( .NOT. (((0 .EQ.
ISIGN) .OR. (0 .EQ. 1-ISIGN)) .AND.
(0 .NE. -IOP+NE))) .AND. (0 .NE.
-1+IOP.EQ) .AND. (0 .LE.
-SPACE(1+LIS)))
 .IMPLIES.
((0 .NE. NE+SPACE(K)) .OR. (0 .LE.
-SPACE(1+LIS)))
```

13

```
COND LIS .NE. 0 .AND. SPACE(K)=LE
20    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
       .NOT. ((ISIGN .EQ. 0 .OR. ISIGN .EQ. 1) .AND. IOP .NE. NE)
       .NOT. (IOP.EQ .LT. AND.ISIGN .EQ. -1)
      LIS=SPACE(LIS+1)
       .NOT. (LIS .LE. 0)
COND LIS .NE. 0 .AND. SPACE(K)=LE
```

```
((0 .NE. -LIS) .AND. (0 .EQ.
LE-SPACE(K)) .AND. ( .NOT. (((0 .EQ.
ISIGN) .OR. (0 .EQ. 1-ISIGN)) .AND.
(0 .NE. -IOP+NE))) .AND. (0 .NE.
-1+IOP.EQ) .AND. (0 .LT.
SPACE(1-LIS)))
 .IMPLIES.
((0 .NE. -SPACE(1+LIS)))
```

14

```
COND LIS .NE. 0 .AND. SPACE(K)=IEQ
30    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
      (ISIGN .EQ. 0 .AND. IOP .EQ. IEQ)
200   FAIL=0
COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0
```

```
((0 .NE. -LIS) .AND. (0 .EQ.
IEQ-SPACE(K)) .AND. ((0 .EQ. ISIGN)
.AND. (0 .LE. IEQ-IOP)))
 .IMPLIES.
((0 .NE. -IOP+NE) .AND. (0 .EQ.
1-ISIGN) .AND. (0 .EQ. LT-SPACE(K)))
```

```
15
      COND LIS .NE. 0 .AND. SPACE(K)=IEQ
30    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
       .NOT. (ISIGN .EQ. 0 .AND. IOP .EQ. IEQ)
      LIS=SPACE(LIS+1)
      (LIS .LE. 0)
100   FAIL=1
      COND (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,SPACE(K) .NE. NE)
      COND .OR. (LIS .LT. 0)
                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                    IEQ-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                    ISIGN) .AND. (0 .EQ. IEQ-IOP)))
                                                    .AND. (0 .LE. -SPACE(1+LIS)))
                                                     .IMPLIES.
                                                    ((0 .NE. NE+SPACE(K)) .OR. (0 .LE.
                                                    -SPACE(1+LIS)))

16
      COND LIS .NE. 0 .AND. SPACE(K)=IEQ
30    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
       .NOT. (ISIGN .EQ. 0 .AND. IOP .EQ. IEQ)
      LIS=SPACE(LIS+1)
       .NOT. (LIS .LE. 0)
      COND LIS .NE. 0 .AND. SPACE(K)=IEQ
                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                    IEQ-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                    ISIGN) .AND. (0 .EQ. IEQ-IOP)))
                                                    .AND. (0 .LT. SPACE(1-LIS)))
                                                     .IMPLIES.
                                                    ((0 .NE. -SPACE(1+LIS)))

17
      COND LIS .NE. 0 .AND. SPACE(K)=NE
40    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
      (IOP .EQ. IEQ .AND. (ISIGN .EQ. -2 .OR. ISIGN .EQ. 1))
200   FAIL=0
      COND SPACE(K)=LT .AND. ISIGN=1 .AND. IOP .NE. NE .AND. FAIL=0
                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                    NE-SPACE(K)) .AND. ((0 .EQ. IEQ-IOP)
                                                    .AND. ((0 .EQ. -2-ISIGN) .OR. (0
                                                    .EQ. 1-ISIGN))))
                                                     .IMPLIES.
                                                    ((0 .NE. -IOP+NE) .AND. (0 .EQ.
                                                    1-ISIGN) .AND. (0 .EQ. LT-SPACE(K)))

18
      COND LIS .NE. 0 .AND. SPACE(K)=NE
40    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
       .NOT. (IOP .EQ. IEQ .AND. (ISIGN .EQ. -2 .OR. ISIGN .EQ. 1))
```

```
      LIS=SPACE(LIS+1)
      (LIS .LE. 0)
100   FAIL=1
COND (SPACE(K) .NE. LT,SPACE(K) .NE. LE,SPACE(K) .NE. IEQ,SPACE(K) .NE. NE)
COND .OR. (LIS .LE. 0)
                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                    NE-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                    IEQ-IOP) .AND. ((0 .EQ. -2-ISIGN)
                                                    .OR. (0 .EQ. 1-ISIGN)))) .AND. (0
                                                    .LE. -SPACE(1+LIS)))
                                                     .IMPLIES.
                                                    ((0 .NE. NE+SPACE(K)) .OR. (0 .LE.
                                                    -SPACE(1+LIS)))

19
COND LIS .NE. 0 .AND. SPACE(K)=NE
40    LOOK=SPACE(LIS)
      CALL DIFF(IREL,LOOK,ISIGN,IOP)
       .NOT. (IOP .EQ. IEQ .AND. (ISIGN .EQ. -2 .OR. ISIGN .EQ. 1))
      LIS=SPACE(LIS+1)
       .NOT. (LIS .LE. 0)
COND LIS .NE. 0 .AND. SPACE(K)=NE
                                                    ((0 .NE. -LIS) .AND. (0 .EQ.
                                                    NE-SPACE(K)) .AND. ( .NOT. ((0 .EQ.
                                                    IEQ-IOP) .AND. ((0 .EQ. -2-ISIGN)
                                                    .OR. (0 .EQ. 1-ISIGN)))) .AND. (0
                                                    .LT. SPACE(1-LIS)))
                                                     .IMPLIES.
                                                    ((0 .NE. -SPACE(1+LIS)))
```

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)                    MAIN    2
      COMMON/GLOBAL/SPACE(14000),FAIL,FREE/DIGOP/DIGITS,OP,ICOM,ISEMI,       BIGGER  1
     1ICOL/ARRAYS/IHEAD(130)/CON/ICOND/ALPHNO/Q(63)                          MAIN    4
      INTEGER COMPARE                                                        MAIN    5
      INTEGER Q,PROG(9),FAIL,FREE,XX,PROG,SPACE,X,DIGITS,OP,PWONG(80)        MAIN    6
      DATA IHEAD/4HSTAT,4HEMEN,4HTS R,4HEMAI,4HNING,4H TO ,4HBE V,4HERIF     MAIN    7
     1 ,4HIED /                                                              MAIN    8
      CALL INITIA                                                            MAIN    9
      CALL ALIST(DIGITS,Q(27))                                               MAIN   10
      DO 85 I=1,9                                                            MAIN   11
      ITZERO=Q(27+I)                                                         MAIN   12
   85 CALL INSERT(DIGITS,0,ITZERO)                                           MAIN   13
      CALL ALIST(OP,Q(37))                                                   MAIN   14
      CALL INSERT(OP,1,Q(39))                                                MAIN   15
      CALL INSERT(OP,2,Q(39))                                                MAIN   16
      CALL INSERT(OP,3,Q(41))                                                MAIN   17
      CALL INSERT(OP,4,Q(42))                                                MAIN   18
      CALL INSERT(OP,5,Q(40))                                                MAIN   19
      CALL ALIST(ICOM,Q(46))                                                 MAIN   20
      CALL ALIST(ISEMI,Q(63))                                                MAIN   21
      CALL ALIST(ICOL,Q(51))                                                 MAIN   22
      CALL ALIST(ICOND,Q(4))                                                 MAIN   23
      CALL INSERT(ICOND,0,Q(14))                                             MAIN   24
      CALL INSERT(ICOND,0,Q(15))                                             MAIN   25
      CALL INSERT(ICOND,0,Q(3))                                              MAIN   26
      WRITE(6,31)                                                            MAIN   27
      JUMP=0                                                                 MAIN   28
      X=0                                                                    MAIN   29
C     PRINT                                                                  MAIN   30
      WRITE(5,205)PROG                                                       MAIN   31
      IF(PWONG(1).EQ.Q(5))GO TO 23                                           MAIN   32
      IF(PWONG(1).EQ.Q(6))GO TO 1000                                         MAIN   33
  205 FORMAT(9A1)                                                            MAIN   34
      WRITE(6,102)PROG                                                       MAIN   35
  102 FORMAT(1X,9A1)                                                         MAIN   36
      IF(PWONG(1).EQ.Q(3).AND.(PWONG(2).NE.Q(15).OR.PWONG(3).NE.Q(14)        MAIN   37
     1 .AND.PWONG(2).NE.Q(4)))GO TO 100                                      MAIN   38
      DO 104 I=1,80                                                          MAIN   39
      IF(PWONG(81-I)-Q(45))105,103,105                                       MAIN   40
  103 CONTINUE                                                               MAIN   41
      GO TO 106                                                              MAIN   42
  105 LAST=I                                                                 MAIN   43
      DO 107 I=LAST,80                                                       MAIN   44
      CALL INSERT(X,0,PWONG(81-I))                                           MAIN   45
  104 CONTINUE                                                               MAIN   46
      IPK12=PROG                                                             CORRED  1
      CALL CONCAT(PROG,IPK12,X)                                              MAIN   58
      CALL RESET(X)                                                          MAIN   59
      CALL CONCAT(PROG,PROG,ISEMI)                                           MAIN   60
      GO TO 100                                                              MAIN   61
   23 WRITE(6,23)                                                            MAIN   62
   23 FORMAT(1X,24(1H-))                                                     MAIN   63
      PROG = CLEAR(PROG)                                                     MAIN   64
```

```
0001B5    30    FORMAT(//)                                          MAIN      65
0001B5          CALL INFX(PROG)                                     MAIN      66
0001B9          WRITE(6,21)                                         MAIN      67
0001C2    31    FORMAT(1H1)                                         MAIN      68
0001C2          CALL REDUCE(PROG)                                   MAIN      69
0001C4          CALL ERPRT(PROG)                                    MAIN      70
0001C5          CALL ERMSG(PROG)                                    MAIN      71
0001C8          IF(FAIL.EQ.1)GO TO 987                              MAIN      72
0001D2          WRITE(6,31)                                         MAIN      73
0001D6          IMAX=0                                              MAIN      74
0001D7          N=0                                                 MAIN      75
0001D9          CALL COPPY(IPROG,PROG,0)                            MAIN      76
0001E2          CALL SMOST(IPROG)                                   MAIN      77
0001E4          I=IPROG                                             MAIN      78
0001E6    42    IF(SPACE(I).EQ.0(63))GO TO 41                       MAIN      79
0001F1          I=SPACE(I+1)                                        MAIN      80
0001F3          N=N+1                                               MAIN      81
0001F4          GO TO 42                                            MAIN      82
0001F6    41    IF(IMAX.LT.N)IMAX=N                                 MAIN      83
0001F7          N=0                                                 MAIN      84
0001F9          I=SPACE(I+1)                                        MAIN      85
0001FC          IF(I.EQ.0)GO TO 42                                  MAIN      86
000203          IF(IMAX.GT.80)IMAX=80                               MAIN      87
000206          CALL MOVETI(IPROG)                                  MAIN      88
000209          WRITE(6,50)                                         MAIN      89
000214    50    FORMAT(5H PATH,16(1H ),4HPATH)                      MAIN      90
000214          IMAX=IMAX+15                                        MAIN      91
000216          JMAX=IMAX                                           MAIN      92
000219          IMAX=IMAX/4                                         MAIN      93
000220          DO 51 IXX=1,IMAX                                    MAIN      94
000221    51    IHEAD(IXX)=0(45)                                    MAIN      95
000225          IXX=IMAX+1                                          MAIN      96
000226          IMAX=IMAX+4                                         MAIN      97
000230          DO 53 IX = IXX,IMAX                                 MAIN      98
000232          JX=IX-IXX+1                                         MAIN      99
000234    53    IHEAD(IX)=IHEAD(JX)                                 MAIN     100
000271          WRITE(6,52)(37),(IHEAD(IXX),IXX=1,IMAX              )MAIN     101
000305    52    FORMAT(31,3244)                                     MAIN     102
000305          WRITE(6,45)                                         MAIN     103
000311    46    FORMAT(1X,>NUMBER)                                  MAIN     104
000311          KK=130-JMAX                                         MAIN     105
000313          CALL FIND(P,S,JMAX,KK)                              MAIN     106
000315          GO TO 947                                           MAIN     107
000317    1000  STOP                                                MAIN     108
000321          END                                                 MAIN     109
```

```
                      SUBROUTINE FIND(SLOCST,JJJX,KK)                                    FIND      2
C000005               INTEGER SLOCST,COND,CONE,DOWNAR,EACH,ENDST,F,FAIL,PAT1,PAT3,PROGA,  FIND      3
                     1 SLOC,SLOCST,SPACE,X,Y,BLANK,SUP,RETX,DOWSEM,RGOTO,Q                FIND      4
                     2,VERSION,SIMPLI                                                    FIND      5
000006               COMMON/BLOCK3/DOWNAR,BLANK,IF,SUP,RETX,PAT1,PAT3,DOWSEM,            FIND      6
                     1SYMTAB/ALOCK/SPACE( 14000),FAIL,IFQQ/DIGOP/ID141,                  BIGGER    2
                     2ID142,ID143,SEMICO,CONE/CON/COND/ALPHNU/Q(63)                      FIND      8
000015               COMMON/FLG/JFLG                                                     COR8D     1
                     3/TABLE/JJJ(500),KKK(100,2)                                         COR8      1
000016               DATA IPROG/2HTR/                                                    COR5      1
000025               Q=0                                                                 FIND      9
000027               JFLG=JJJ(1)                                                         COR8D     2
000031               K=0                                                                 FIND     10
000011               CII=0                                                               FIND     11
000012               IF=0                                                                FIND     12
000013               I=0                                                                 FIND     13
000014               F=0                                                                 FIND     14
000015               PROGA=0                                                             FIND     15
000016               IPROGP=0                                                            CORBP     1
000017               CALL NLIST(PAT1,Q( 9))                                              FIND     16
000021               CALL INSERT(PAT1,1,Q( 6))                                           FIND     17
000024               CALL INSERT(PAT1,2,Q(41))                                           FIND     18
000027               CALL NLIST(DOWSEM,Q(57))                                            FIND     19
000032               CALL NLIST(PAT3,Q( 7))                                              FIND     20
000035               CALL INSERT(PAT3,1,Q(15))                                           FIND     21
000040               CALL INSERT(PAT3,2,Q(45))                                           FIND     22
000043               CALL INSERT(PAT3,3,Q(20))                                           FIND     23
000046               CALL INSERT(PAT3,4,Q(15))                                           FIND     24
000051               CALL INSERT(PAT3,5,Q(45))                                           FIND     25
                C    SPACE = 4H                                                          FIND     26
                C    PAT1 = 2HIF(                                                        FIND     27
                C    COND = 2HC)                                                         FIND     28
                C    NUMBER = 10H0123456789                                              FIND     29
                C    PROGA = 3HPRO                                                       FIND     30
                C    BLANK = 1H                                                          FIND     31
                C    CONE = 1H=                                                          FIND     32
                C    PAT3 = 4HGO TO                                                      FIND     33
000056               CALL NLIST(BLANK,Q(45))                                             FIND     34
000057               CALL NLIST(IF,Q( 9))                                                FIND     35
000062               CALL INSERT(IF,1,Q( 6))                                             FIND     36
000065               CALL NLIST(SUP,Q(19))                                               FIND     37
000071               CALL INSERT(SUP,1,Q(20))                                            FIND     38
000074               CALL INSERT(SUP,2,Q(15))                                            FIND     39
000077               CALL INSERT(SUP,3,Q(15))                                            FIND     40
000102               CALL NLIST(RETX,Q(15))                                              FIND     41
000105               CALL INSERT(RETX,1,Q( 5))                                           FIND     42
000107               CALL INSERT(RETX,2,Q(20))                                           FIND     43
000112               CALL INSERT(RETX,3,Q(21))                                           FIND     44
000115               CALL INSERT(PLIX,4,Q(18))                                           FIND     45
000120               CALL INSERT(PLIX,5,Q(14))                                           FIND     46
000123               CALL NLIST(DOWSEM,Q(57))                                            FIND     47
000126               CALL INSERT(DOWSEM,1,Q(63))                                         FIND     48
000131               CALL COPY(SLOCST,PAT3,Q)                                            FIND     49
```

```
                      CALL INSERT(SGOTO,0,0(42))                                FIND      50
000137                CALL BLIST(IN,0(53))                                      COR10      1
                    C*****************************************                 FIND      51
                    C     PATX = # #                                          FIND      52
                    C     CONCAT = A,#                                        FIND      53
                    C     BLANK= # #                                          FIND      54
                    C     CLOSE = #;#                                         FIND      55
                    C     IF  = #IF#                                          FIND      56
                    C     STOP = #STOP#                                       FIND      57
                    C     RETX= #RETURN#                                      FIND      58
                    C     END= #END#                                          FIND      59
                    C     IFF= #IF(#                                          FIND      60
                    C     DOWSEMI=#;#                                         FIND      61
                    C     PGOTO = #)GO TO #                                   FIND      62
000142                IF(SPACE(SLOCST).EQ.0(3))GO TO 4                          FIND      63
000146                CALL INSERT(SLOCST,0,0(63))                               FIND      64
000151                CALL INSERT(SLOCST,0,ITRUE)                               COR5       2
000155                IPK12=SLOCST                                              FIND      71
000160                CALL CONCAT(SLOCST,COND,IPK12)                            FIND      72
000162    4           CALL BREAK(SLOCST,SEMICO,ENDST)                           FIND      73
000166    1           CALL COPY(LINE,SLOCST,ENDST)                              FIND      74
000173                ENDST=SPACE(ENDST+1)                                      FIND      75
000175                IPK12=SLOCST                                              FIND      76
000200                CALL DELETE(SLOCST,IPK12,ENDST)                           FIND      77
000202    1001        IPK12=PROGA                                               FIND      78
000204                CALL CONCAT(PROGA,IPK12,DOWNAR )                          FIND      79
000206                CALL MLIST(ITEMP,N)                                       FIND      80
000210                CALL CONS(PROGA,ITEMP,DOWNAR,LINE,DOWNAR,ITEMP,DOWNAR,SEMICO)  COP4       1
000220                CALL DELETE(ITEMP)                                        FIND      88
000222    1002        CALL BREAK(SLOCST,SEMICO,ENDST)                           FIND      89
000224                IF(FAIL) 13,14,1003                                       FIND      90
000232    14          CALL COPY(LINE,SLOCST,ENDST)                              FIND      91
000235                ENDST=SPACE(ENDST+1)                                      FIND      92
000237                IPK12=SLOCST                                              FIND      93
000241                CALL DELETE(SLOCST,IPK12,ENDST)                           FIND      94
000243                CALL FIND(SCLIPS,COMP,X,Y,1)                              FIND      95
000250                IF (-X1L) 1001,1001,33                                    FIND      96
000254    33          X=X+1                                                     FIND      97
000256    1009        IF(SPACE(LINE).EQ.0(43))GO TO 1009                        FIND      98
000261                CALL LEN(LINE,7,IPT)                                      FIND      99
000265                CALL SPLIT(IPT,PAT1,BEGST,ENDST,1)                        FIND     100
000273                IF(FAIL) 2000,2000,1004                                   FIND     101
000275    2000        CALL SPLIT(IPT,RSUTO,BEGST,ENDST,0)                       FIND     102
000277                IF(FAIL) 2004,2004,1009                                   FIND     103
000303    2004        CALL INSERT(LINE,9,0(51))                                 FIND     104
000305                CALL LEN(LINE,9,K)                                        FIND     105
000311                CALL INSERT(LINE,9,0(51))                                 FIND     106
000314                K=K+1                                                     FIND     107
000316    1004        IPK12=PROGA                                               FIND     108
000320                CALL CONCAT(PROGA,IPK12,LINE)                             FIND     109
000322                CALL CONCAT(PROGA,PROGA,SEMICO)                           FIND     110
000325    1007        CONTINUE                                                  FIND     111
000330                CALL BREAK(SLOCST,SEMICO,ENDST)                           FIND     112
000341                IF(FAIL) 77,77,1003                                       FIND     113
```

```
000335   77    CALL COPY(LINE,SLOCST,ENDST)                    FIND    114
000340         ENDST=SPACE(ENDST+1)                            FIND    115
000342         IPK12=SLOCST                                    FIND    116
000345         CALL DELETE(SLOCST,IPK12,ENDST)                 FIND    117
000347         CALL STRINGS(LINE,COND,X,Y,1)                   FIND    118
000353         IF(FAIL)1001,1001,1006                          FIND    119
000361   1003    N=0                                           FIND    120
000361         N=0                                             FIND    121
000362         INDICA=0                                        FIND    122
000363   1010  F=SRCH(1,PROGA,INDICA,LOOPER,NPLOOP,N)          CORBP     2
000370         IF(FAIL.EQ.1) GOTO 1111                         FIND    124
000374         IF(INDICA.EQ.0)GO TO 6                          FIND    125
000375   3     CALL ANY(F,DIRNAM,KA,0)                         FIND    126
000403         IF(FAIL.EQ.1)GO TO 2                            FIND    127
000404         CALL LEN(KA,3,KB)                               FIND    128
000407         CALL DELETE(F,KA,KB)                            FIND    129
000412         GO TO 3                                         FIND    130
000415   2     IPK12=F                                         FIND    131
000417         N=N+1                                           FIND    132
000420         WRITE(6,30)N                                    FIND    133
000426   30    FORMAT(//,I4)                                   FIND    134
000426   5     CALL BREAK(IPK12,SEMICO,KA)                     FIND    135
000431         CALL COPY(LINE,IPK12,KA)                        FIND    136
000434         CALL JUSTPT(LINE,9,80)                          FIND    137
000437         IPK12=SPACE(KA+1)                               FIND    138
000441         IPK12=SPACE(IPK12+1)                            FIND    139
000443         IF(SPACE(IPK12).NE.J(51))GO TO 5                FIND    140
000447         CALL COPPY(LYNX,F,IPK12)                        FIND    141
000451         CALL DELETE(F)                                  FIND    142
000453         ISTR=0                                          CORBC     1
000454         IF(JFLG.EQ.1.OR.LOOPER.EQ.0)GO TO 203           CORBD     3
000465         LP=LOOPER                                       CORB      3
000465         KA=SPACE(LP)                                    CORB      4
000470         IF(KA.EQ.0)GO TO 203                            CORB      5
000471   204   LP=SPACE(LP+1)                                  CORB      6
000473         IF(LP.EQ.0)GO TO 200                            CORB      7
000474         KA=SPACE(LP)                                    CORB      8
000475         IF(KA.EQ.0)GO TO 203                            CORB      9
000476         IF(SPACE(KA).EQ.J(51))GO TO204                  CORB     10
000500         CALL COPPY(KF,KA,0)                             COR10     2
000502         CALL LAST(KF)                                   COR10     3
000504         CALL STRING(KA,KF,K1,K2,1)                      COR10     4
000511         CALL DELETE(KF)                                 COR10     5
000512         IF(FAIL.EQ.1)GO TO 204                          COR10     6
000514         K2=SPACE(K2+1)                                  COR10     7
000520         IF(SPACE(K2).EQ.J(42))GO TO 204                 COR10     8
000522         IF(SPACE(K2).NE.J(43))GO TO 204                 COR10     9
000524         KA=KA                                           CORB     13
000524         GO TO 204                                       CORB     14
000526   203   ISTR=0                                          CORB     15
000527         IF(JFLG.EQ.0)GO TO 202                          CORBP     3
000530         NPLOOP=NPLOOP-1                                 CORBP     4
000532         GO TO 202                                       CORB     16
000532   200   CALL COPPY(ISTR,KA,0)                           CORB     17
```

```
000535   202   LYNX=VEXC[illegible](LYNX,ISTF,SPLCUP)                   COR8P      5
000541         LYNX=SIMPLI(LYNX)                                       FIND     144
000543         KB=LYNX                                                 COR10     10
000544   207   CALL ANY(KB,IM,KA,0)                                    COR10     11
000547         IF(FAIL.EQ.1)GO TO 205                                  COR10     12
000553         CALL CAL2(KA,0)                                         COR10     13
000555         IF(FAIL.EQ.0)GO TO 205                                  COR10     14
000560         KA=SPACE(KA+1)                                          COR10     15
000562         GO TO 207                                               COR10     16
000563   205   CALL COPPY(KB,LYNX,KA)                                  COR10     17
000566         CALL JOUTPT(KB,JMAX,KK)                                 COR10     18
000572         CALL DELET3(KB)                                         COR10     19
000574         CALL COPPY(KB,KA,SPACE(KA+1))                           COR10     20
000600         CALL JOUTPT(KB,JMAX,KK)                                 COR10     21
000604         CALL DELET3(KB)                                         COR10     22
000606         CALL JOUTPT(SPACE(KA+1),JMAX,KK)                        COR10     23
000613         CALL DELET3(LYNX)                                       COR10     24
0006[illegible]         GO TO 1010                                              COR10     25
000620   205   CALL JOUTPT(LYNX,JMAX,KK)                               COR10     26
000622         CALL DELET3(LYNX)                                       FIND     146
000624         GO TO 1010                                              FIND     147
000627         [illegible]+1                                           FIND     148
0006[illegible]         GO TO 1010                                              FIND     149
000631   1111  CONTINUE                                                FIND     150
000631         CALL DELET3([illegible])                                FIND     151
000633         CALL DELET3(BLANK)                                      FIND     152
000635         CALL DELET3(IP)                                         FIND     153
000637         CALL DELET3(SUP)                                        FIND     154
000641         CALL DELET3([illegible])                                FIND     155
000643         CALL DELET3([illegible]SEA)                             FIND     156
000645         CALL DELET3([illegible]JTO)                             FIND     157
000647         CALL DELET3(SLACST)                                     FIND     158
000653         CALL DELET3([illegible])                                FIND     159
000655         CALL DELET3(PAT1)                                       FIND     160
000656         CALL DELET3(PSCGA)                                      FIND     161
000660         CALL DELET3(PAT3)                                       FIND     162
000662         CALL DELET3(I[illegible])                               COR10     27
000664         IF(JI[illegible](1).EQ.1)RETURN                         COR8C      3
000671         DO 250 I=1,100                                          COR8      19
000673         IF(IREC(I,2).EQ.0)GO TO 250                             COR8      21
000676         CALL DELET3(IREC(I,2))                                  COR8      22
000676   250   CONTINUE                                                COR8      23
000702         RETURN                                                  FIND     163
000703         END                                                     FIND     164
```

```
              INTEGER FUNCTION EACH(N,SLOCST,INDICA,LOOPER,NPLOOP,NPATH)         CORBP      6
000011        COMMON /BLOCK/SPACE(14000),FAIL,IFQQ/BLOCK3/DOWNAR,BLANK,IF,SUP,   BIGGER     3
             1 [illegible]/DIGOP/DIGIT,                                          EACH       4
             2IS25,IS53,[illegible]/ALP[illegible](63)/UPKEEP/IPAD(500),IREC(100,2)  CORB  25
000011        INTEGER [illegible],FAIL,PREV,A,BEGST,BLANK,CONE,DIGIT,DOWNAR,     EACH       6
             1[illegible],ENDST,GOTO,P,PATH,PATX,STR,STR3,SUP,X,Y,RETX,          EACH       7
             2SLOCST,SLOCST,SPACE,SPP                                            EACH       8
000011        COMMON/FLG/JFLAG                                                   CORBD      4
000011        COMMON/KTAB/ITERM(25,5)                                            CORBP      7
        C     SET LOCAL VARIABLES TO ZERO                                        EACH       9
000011        IF(INDICA.EQ.1)GO TO 1                                             EACH      10
000013        P=0                                                                EACH      11
000014        PATH=0                                                             EACH      12
[illegible]   [illegible]=0                                                      CORB      26
[illegible]   [illegible]                                                        EACH      13
[illegible]   [illegible]                                                        EACH      14
000017        PATX=0                                                             EACH      15
000020        LABEL=0                                                            EACH      16
000021        LINE=0                                                             EACH      17
000022        ITEMP=0                                                            EACH      18
000023           LABEL2=0                                                        EACH      19
000024         STR=0                                                             EACH      20
[illegible]      [illegible]=0                                                   EACH      21
[illegible]    [illegible]                                                       EACH      22
000027        CALL [illegible]LIST(PATX,0)                                       EACH      23
[illegible]   CALL INSERT(PATX,0,Q(57))                                          EACH      24
[illegible]   CALL CONCAT(PATX,PATX,DOWNAR)                                      EACH      25
[illegible]   CALL SE[illegible](SLOCST,PATX,P,Y,0)                              EACH      26
[illegible]   CALL [illegible](P,[illegible],0)                                  EACH      27
[illegible]   CALL [illegible],DOWNAR,JPP,0)                                     EACH      28
[illegible]   IF(FAIL.EQ.1)GO TO 6                                               EACH      29
[illegible]   CALL [illegible]PY([illegible],P,[illegible])                      FACH      30
[illegible]   [illegible]                                                        EACH      31
[illegible]   CALL [illegible]AT(PATH,IPK12,LINE)                                EACH      32
[illegible]   [illegible]                                                        EACH      33
000074        IF(JFLAG.EQ.1)GO TO 801                                            CORBC      5
000102        [illegible]=0                                                      CORB      27
000103        DO 3 I=1,100                                                       CORB      28
000104        IF (IREC(I,1).GE.0)GO TO 3                                         CORB      29
[illegible]   IF([illegible].EQ.0)GO TO 5                                        CORB      30
000107        [illegible]+1                                                      CORB      31
000111   3    CONTINUE                                                           CORB      32
000113   5    [illegible]=1                                                      CORB      33
000115   801  IF(SPACE(P).EQ.Q(43))GO TO 11                                      CORBC      6
000120        CALL [illegible](P,7,JPP)                                          EACH      35
000122        IF(FAIL.EQ.1)GO TO 6                                               EACH      36
        C  IF IT IS A STOP GO TO 134                                             EACH      37
000130        CALL ST[illegible](JPP,SUP,BEGST,ENDST,1)                          EACH      38
000133        IF(FAIL.EQ.1)GO TO 4                                               EACH      39
000141        [illegible]ST=SPACE(ENDST+1)                                       EACH      40
000142        IF(SPACE([illegible]ST).NE.Q(43))GO TO 4                           EACH      41
000145        GO TO 134                                                          EACH      42
        C  IF IT IS A [illegible] GO TO 134                                      EACH      43
```

```
000145    4     CALL STRING(JPP,RETX,BEGST,ENDST,1)                    EACH     44
000151          IF(FAIL.EQ.1)GO TO 7                                   EACH     45
000157          ENDST=SPACE(ENDST+1)                                   EACH     46
000160          IF(SPACE(ENDST).NE.0(63))GO TO 7                       EACH     47
000163          GO TO 134                                              EACH     48
          C IF IT IS AN END SET FAIL TO 1 AND GO TO 133                EACH     49
000166    6     FAIL=1                                                 EACH     50
000170          GO TO 133                                              EACH     51
000172    7     IF(SPACE(P).EQ.0(43))GOTO  11                          EACH     52
000175          CALL LAV(P,7,JPP)                                      EACH     53
          C IF IT IS NOT IF( GO TO 1127                                EACH     54
000177          CALL STRING(JPP,IFF,BEGST,ENDST,1)                     EACH     55
000203          IF(FAIL) 10,10,1127                                    EACH     56
000206    10    JPP=SPACE(ENDST+1)                                     EACH     57
          C IF IT DOES MATCH )GO TO  THEN JUMP TO 11                   EACH     58
000211          CALL STRING(JPP,RGHT),BEGST,ENDST,0)                   EACH     59
000216          IF(FAIL.EQ.1)GO TO 11                                  EACH     60
000221          CALL COPY(ITPT,P,BEGST)                                EACH     61
000225          IF(JMAG.EQ.1)GO TO 27                                  CORBC     7
000230          CALL COPY(ITPT,IREC(NPTR,2),0)                         CORB     34
000235          CALL INSERT(LOOPER,0,ITPT)                             CORB     35
000241          CALL PLIST(ITPT,NPTR)                                  CORB     36
000245          CALL INSERT(ITPT,2,0(61))                              CORB     37
000252          CALL INSERT(LOOPER,2,0,ITPT )                          CORB     38
          C LINE IS ASSERTIONS UP TO AND INCLUDING )                   EACH     62
          C GO ON TO 27                                                EACH     63
000257          GOTO 27                                                EACH     64
          C IF IT IS NOT A GO TO  JUMP TO 11                           EACH     65
000264    1127  CALL STRING(JPP,GOTO,BEGST,ENDST,1)                    EACH     66
000270          IF(FAIL.EQ.1)GO TO 11                                  EACH     67
000274          CALL DELETE(IFF)                                       EACH     68
000277    27    SPP=SPACE(ENDST+1)                                     EACH     69
000301          CALL CLEAN(SPP,SPACE,ENDST)                            EACH     70
000306          CALL COPY(LABEL,SPP,ENDST)                             EACH     71
          C  LABEL IS THE LABEL OF THE NEST STMT TO BE                 EACH     72
000313          GOTO 591                                               EACH     73
          C  IF NOT IF THE ABOVE HOLDS CONCATENATE LIN                 EACH     74
000316    11    CALL APP(P,SPACE,ENDST,0)                              EACH     75
000324          CALL COPY(LINE,P,ENDST)                                EACH     76
000331          P=SPACE(ENDST+1)                                       EACH     77
000333          CALL FORMAT(PATH,PATH,LINE)                            EACH     78
000345          IF(JMAG.EQ.1)GO TO 592                                 CORBC     8
000350          CALL COPY(ITPT,IREC(NPTR,2),0)                         CORB     39
000361          CALL INSERT(LOOPER,0,ITPT)                             CORB     40
          C THEN IF  POST STMT IS NOT GOTO GO T 7                      EACH     79
000367    592   CALL STRING(P,ORPAR,BEGST,ENDST,1)                     EACH     80
          C NOT IF IT IS THE LAST ASSERTION,G) TO 6                    EACH     81
000403          IF(P.EQ.0) GO TO 6                                     EACH     82
          C  THEN ADD ) TO 592                                         EACH     83
000406          IF(JMAG.EQ.1)NPTR=NPTR+1                               CORBC     9
000412          IF(FAIL)593,593,7                                      EACH     84
          C  AT THIS POINT IT IS EITHER GO TO OR IF()GO                EACH     85
000416    591   CALL STRING(LINE,IF,BEGST,ENDST,0)                     EACH     86
000423          IF(FAIL) 21,2175,4                                     EACH     87
```

```
         CIF DELIMITER DELETE THE IF AND CONCAT REMAINDE             EACH    88
003376   21    CALL DELETE(LINE,BEGST,ENDST)                         EACH    89
00[illegible]       IPK12=PATH                                        EACH    90
000403         CALL CONCAT(PATH,IPK12,LINE)                          EACH    91
000405         CALL CONCAT(PATH,PATH,SEMICO)                         EACH    92
         C CREATE [illegible] SEE IF IT MATCHES AN                   EACH    93
000410   5[illegible]  CALL CONCAT(ITEMP,DOUSEM,LABEL)               EACH    94
000413         CALL CONCAT(ITEMP,ITEMP,BLANK)                        EACH    95
         C IF NOT GO TO 25                                           EACH    96
000416         CALL STRING(SLOCST,ITEMP,BEGST,ENDST,0)               EACH    97
000425         IF(FAIL.EQ.0)GO TO 26                                 EACH    98
000432         CALL INSERT(ITEMP,2,O(43))                            EACH    99
000436         CALL STRING(SLOCST,ITEMP,BEGST,ENDST,0)               EACH   100
000445         IF(FAIL.EQ.1)GO TO 25                                 EACH   101
000453   26    J=PREV(SLOCST,BEGST)                                  EACH   102
000461         M=SPACE(M)                                            EACH   103
         C IF SO M IS THE NUMBER OF THE COND AND GO TO               EACH   104
00[illegible]       GO TO 595                                         EACH   105
00[illegible]3 25    CALL DELETE(ITEMP,0)                            EACH   106
         C  TRY IT WITHOUT THE DA                                    EACH   107
000465         CALL STRING(SLOCST,ITEMP,BEGST,ENDST,0)               EACH   108
000474         IF(FAIL.EQ.1)GO TO 28                                 EACH   109
00[illegible]       CALL DELETE(ITEMP,1)                              EACH   110
00[illegible]       CALL STRING(SLOCST,ITEMP,BEGST,ENDST,0)           EACH   111
00[illegible]  28    M=SPACE(BEGST+1)                                EACH   112
         C M POINTS TO THE STATEMENT WITH THAT LABEL A               EACH   113
00[illegible]       IF(FLAG.EQ.0)NPTR=IABS(IREC(NPTR,1))              CORBC   10
00[illegible]       GO TO 7                                           EACH   114
00[illegible]  595   CALL DELETE(STR)                                EACH   115
         [illegible] CONDITION M                                     EACH   116
00[illegible]       CALL GETST(STR,M)                                 EACH   117
00[illegible]       CALL INSERT(STR,0,O(57))                          EACH   118
00[illegible]       CALL CONCAT(STR,STR,DOWNAS)                       EACH   119
00[illegible]       CALL STRING(SLOCST,STR,BEGST,ENDST,0)             EACH   120
00[illegible]       IF(FLAG.EQ.0)NPTR=IABS(IREC(NPTR,1))              CORBC   11
00[illegible]  [illegible] CALL [illegible](SLOCST,SEMICO,ENDST,0)   EACH   121
         C COPY [illegible] TO LINE AND CONCATE TO PATH              EACH   122
         CONCATENATE THE PATH TO EACH                                EACH   123
00[illegible]       CALL COPY(LINE,BEGST,ENDST)                       EACH   124
00[illegible]       IF(FLAG.EQ.1)GO TO 802                            CORBC   12
00[illegible]       CALL COPY(ITPT,IREC(NPTR,2),0)                    CORB    44
00[illegible]       CALL [illegible](ITPT,0,ITPT)                     CORB    45
00[illegible]  802   CALL CONCAT(PATH,PATH,LINE)                     CORBC   13
00[illegible]       CALL COPY(EACH,PATH,0)                            EACH   126
00[illegible]       CALL CONCAT(EACH,EACH,CONE)                       EACH   127
00[illegible]       [illegible]                                       EACH   128
00[illegible]  577   [illegible]                                     EACH   129
00[illegible]  [illegible] CALL [illegible](ISS,[illegible],BEGST,0) EACH   130
00[illegible]       IF(FAIL.EQ.1)GO TO 2                              EACH   131
00[illegible]  5274  CALL [illegible](BEGST,0,ENDST)                 EACH   132
00[illegible]       IF(FAIL.EQ.1)GO TO 2                              EACH   133
00[illegible]       IF(S[illegible](ENDST) - O(51)) 5283,5282,5283   EACH   134
00[illegible]  5283  I=[illegible](BEGST+1)                          EACH   135
00[illegible]       GO TO [illegible]                                 EACH   136
```

```
000455    5282  CALL DELETE(EACH,REGST,ENDST)                    EACH    137
000460          GO TO 5555                                       EACH    138
000464    2     FAIL=0                                           EACH    139
000465          IF(JFLAG.EQ.1)RETURN                             COR8P     8
000471          NPLOOP=NPLOOP+1                                  COR8P     9
000473          ITEM4(NPLOOP,1)=NPATH+1                          COR8P    10
000474          ITEM4(NPLOOP,2)=N                                COR8P    11
000476          ITEM4(NPLOOP,3)=M                                COR8P    12
000477          RETURN                                           EACH    140
000501    1     MARK=0                                           EACH    141
000502          CALL ANY(PATH,SEMICO,KA,0)                       EACH    142
000505    13    P=SPACE(KA+1)                                    EACH    143
000507          IF(P.EQ.0)STOP                                   EACH    144
000510          IF(SPACE(P).EQ.0(43))GO TO 24                    EACH    145
000521          IF(SPACE(P).EQ.0(57))GO TO 1000                  EACH    146
000525          CALL ANY(P,7,KA)                                 EACH    147
000535          IF(SPACE(KA).EQ.0(41))GO TO 583                  EACH    148
000536    24    CALL ANY(P,SEMICO,KA,0)                          EACH    149
000547          GO TO 13                                         EACH    150
000553    583   KA=KA                                            EACH    151
000555          GO TO 24                                         EACH    152
000556    1000  IF(MARK.EQ.0)GO TO 134                           EACH    153
000564          CALL COPY(IPATH,PATH,MARK)                       EACH    154
000571          CALL ANY(MARK,SEMICO,KA,0)                       EACH    155
000574          CALL COPY(IPATH,MARK,SPACE(KA+1))                EACH    156
000611          CALL INSERT(IPATH,3,0(62))                       EACH    157
000626          CALL CONCAT(PATH,IPATH,JPATH)                    EACH    158
000711          CALL DELETE(IPATH)                               EACH    159
000713          CALL ANY(JPATH,COMMA,KA,0)                       EACH    160
000717          CALL LEN(STR,KB)                                 EACH    161
001011          CALL COPY(STR,KB,KA)                             EACH    162
001015          CALL DELETE(IPATH)                               EACH    163
001024          CALL STRIP(SLOCST,STR,KA,KB,0)                   EACH    164
001035          CALL ANY(KA,SEMICO,KA,0)                         EACH    165
001039          KA=KA-(KA+1)                                     EACH    166
001042          IF(JFLAG.EQ.1)GO TO 502                          CORBC    14
001053    695   CALL ANY(SPACE(LOOPER),COMMA,KA2,1)              COR8A     3
001065          IF(FAIL.EQ.1)GO TO 694                           COR8     47
001073          K=SPACE(LOOPER)                                  COR8     48
001075          KK=SPACE(K+1)                                    COR8     49
001107          KPT=SPACE(KK)                                    COR8     50
001051          CALL DELETE(SPACE(LOOPER))                       COR8     51
001075          CALL DELETA(LOOPER,0)                            COR8     52
001103          GO TO 502                                        EACH    167
001127    694   CALL DELETE(SPACE(LOOPER))                       COR8     53
001073          CALL DELETA(LOOPER,0)                            COR8     54
001131          GO TO 5                                          COR8     55
001135    134   FAIL=0                                           EACH    168
001134    133   CALL DELETE(4)                                   EACH    169
001110          CALL DELETE(LINE)                                EACH    170
001112          CALL DELETE(STR3)                                EACH    171
001114          CALL DELETE(STR)                                 EACH    172
001116          CALL DELETE(LABEL2)                              EACH    173
001120          CALL DELETE(LABEL)                               EACH    174
```

```
001122          CALL DELETA(PATH)                                   EACH    175
001124          INDICA=0                                            EACH    176
001130          IF(JFLAG.EQ.1)RETURN                                COR8C    15
001134    693   IF(LOOPER.EQ.0)RETURN                               COR8     56
001137          CALL DELETA(SPACE(LOOPER))                          COR8     57
001143          CALL DELETA(LOOPER,0)                               COR8     58
001151          GO TO 693                                           COR8     59
001155          END                                                 EACH    178
```

```
                      SUBROUTINE ENOUGH(PROG)                                    ENOUGH    2
000002                INTEGER TABS(100,5),BRPAT,SPACE,FAIL,FREE,Q,DIGITS,OP,CYCLIC,PATH,   ENOUGH    3
                     1REC(100,2),CYCLE,FLAG,PROG,OPCODE,RETURN,END,STOP,GOTO,IREC(100,2)   COR8     60
000003                COMMON/IPREP/TABS,IREC                                     COR9     61
000003                COMMON/BLOCK/SPACE(14000),FAIL,FREE/ALPHNU/Q(63)/DIGOP/DIGITS  BTGGER   4
                     1,OP,ICOM,ISEMI,TCONE/CTN/ICOND                            ENOUGH    7
000005                CALL NLIST(RETURN,Q(63))                                   ENOUGH    8
000006                CALL INSERT(RETURN,0,Q(14))                                ENOUGH    9
000011                CALL INSERT(RETURN,0,Q(18))                                ENOUGH   10
000014                CALL INSERT(RETURN,0,Q(21))                                ENOUGH   11
000017                CALL INSERT(RETURN,0,Q(20))                                ENOUGH   12
000022                CALL INSERT(RETURN,0,Q(5))                                 ENOUGH   13
000025                CALL INSERT(RETURN,0,Q(18))                                ENOUGH   14
000030                CALL NLIST(END,Q(63))                                      ENOUGH   15
000033                CALL INSERT(END,0,Q(41))                                   ENOUGH   16
000036                CALL INSERT(END,0,Q(14))                                   ENOUGH   17
000041                CALL INSERT(END,0,Q(5))                                    ENOUGH   18
000044                CALL NLIST(STOP,Q(63))                                     ENOUGH   19
000047                CALL INSERT(STOP,0,Q(16))                                  ENOUGH   20
000052                CALL INSERT(STOP,0,Q(15))                                  ENOUGH   21
000055                CALL INSERT(STOP,0,Q(20))                                  ENOUGH   22
000060                CALL INSERT(STOP,0,Q(10))                                  ENOUGH   23
000063                CALL NLIST(GOTO,Q(45))                                     ENOUGH   24
000066                CALL INSERT(GOTO,0,Q(15))                                  ENOUGH   25
000071                CALL INSERT(GOTO,0,Q(20))                                  ENOUGH   26
000074                CALL NLIST(LOOPL,Q(41))                                    CORR     62
000077                CALL INSERT(LOOPL,0,Q(16))                                 CORR     63
000102                CALL INSERT(LOOPL,0,Q(15))                                 CORR     64
000105                CALL INSERT(LOOPL,0,Q(15))                                 CORR     65
000110                CALL INSERT(LOOPL,0,Q(12))                                 CORR     66
000113                CALL COPY(LOOPS,LOOPL,0)                                   CORR     67
000116                CALL INSERT(LOOPE,0,Q(4))                                  CORR     68
000121                CALL INSERT(LOOPE,0,Q(14))                                 CORR     69
000124                CALL INSERT(LOOPE,0,Q(5))                                  CORR     70
000127                CALL NLIST(JS,Q(63))                                       CORR     71
000132                CALL INSERT(JS,0,Q(55))                                    CORR     72
000135                ISTACK=0                                                   CORR     73
000136                JFLAG=0                                                    CORR     74
000137                DO 17 I=1,100                                              ENOUGH   27
000141                IREC(I,1)=)                                                CORR     75
000142                IREC(I,2)=0                                                CORR     76
000143                DO 17 J=1,5                                                ENOUGH   28
000145    17          TABS(I,J)=0                                                ENOUGH   29
000155                K=1                                                        ENOUGH   30
000156                IPOINT=PROG                                                ENOUGH   31
000157                LINE=0                                                     ENOUGH   32
000160                LAB=0                                                      ENOUGH   33
000161                FLAG=0                                                     ENOUGH   34
000162    19          CALL ANY(IPOINT,ISEMI,KA,0)                                CORR     77
000165                IF(FAIL.EQ.1)GO TO 16                                      CORR     78
000170                CALL STRING(IPOINT,ICOND,KC,KA,1)                          CORR     79
000174                IF(FAIL.EQ.0)GO TO 106                                     CORR     80
000176                CALL COPY(IPOINT,IPOINT,KA)                                CORR     81
```

```
000262          IPOINT=SPACE(KA+1)                                      CORE    82
000263          GO TO 25                                                CORE    83
000265    106   KB=KS                                                   CORE    84
000267    107   KC=SPACE(K)+1)                                          CORE    85
000271          CALL STRING(KC,LOOPL,LA,LB,1)                           CORE    86
000275          IF(FAIL.EQ.1)GO TO 100                                  CORE    87
000220          CALL ANY(L3,JS,LC,0)                                    CORE    88
000223          IF(JFLAG.EQ.1)GO TO 103                                 CORE    89
000226          CALL COPY(LOOPI,LB,LC)                                  CORE    90
000231          CALL INSERT(ISTACK,0,LOOPI)                             CORE    91
000234    108   IF(KA.EQ.LC)GO TO 105                                   CORE    92
000237          CALL DELETE(IPOINT,KC,LC)                               CORE    93
000242          GO TO 107                                               CORE    94
000244    105   IF(K).EQ.K3)GO TO 108                                   CORE    95
000246          CALL DELETE(IPOINT,KC,LC)                               CORE    96
000251          SPACE(K))=)(53)                                         CORE    97
000253          IPOINT=SPACE(KD+1)                                      CORE    98
000255          GO TO 25                                                CORE    99
000256    108   KS=SPACE(KA+1)                                          CORE   100
000260          CALL DELETE(PROG,IPOINT,KA)                             CORE   101
000262          IPOINT=KS                                               CORE   102
000264          GO TO 1)                                                CORE   103
000266    1)3   CALL STRING(KC,LOOPE,LA,LB,1)                           CORE   104
000271          IF(FAIL.EQ.1) GO TO 120                                 CORE   105
000273          CALL ANY(L3,JS,LC,0)                                    CORE   106
000277          IF(JFLAG.EQ.1)GO TO 1)3                                 CORE   107
000302          CALL COPY(LOOPI,LB,LC)                                  CORE   108
000305          CALL INSERT(LOOPI,SPACE(ISTACK))                        CORE   109
000310          IF(FAIL.EQ.1)GO TO 113                                  CORE   110
000312          CALL DELETE(LOOPI)                                      CORE   111
000315          CALL DELETE(SPACE(ISTACK))                              CORE   112
000320          CALL DELETE(ISTACK,0)                                   CORE   113
000322          GO TO 103                                               CORE   114
000324    113   JFLAG=1                                                 CORE   115
000325          GO TO 103                                               CORE   116
000326    121   CALL ANY(KC,JS,LC,0)                                    CORE   117
000331          IF(KS.EQ.LC)GO TO 139                                   CORE   118
000334          KD = LC                                                 CORE   119
000335          GO TO107                                                CORE   120
000337    139   IPOINT=SPACE(KA+1)                                      CORE   121
000341    200   IABS(*,?)=1                                             CORE   122
000341          CALL ANY(IPOINT,T,ISPAT,KA,0)                           ENOUGH  42
000344          IF(FAIL.EQ.1)GO TO 24                                   ENOUGH  43
000347          CALL COPY(LINE,IPOINT,KA)                               ENOUGH  44
000352          IPOINT=SPACE(KA+1)                                      ENOUGH  45
000354    24    IF(SPACE(IPOINT).NE.)(43))GO TO 34                      ENOUGH  46
000357          CALL DELETA(IPOINT,0)                                   ENOUGH  47
000362          CALL INSERT(PROG,7,KA)                                  ENOUGH  48
000365    35    IF(SPACE(KA).EQ.)(6))GO TO 26                           CORE   123
000371          IF(SPACE(KA).NE.)(45))GO TO 34                          ENOUGH  50
000373          KA=SPACE(KA+1)                                          ENOUGH  51
000376          GO TO 35                                                ENOUGH  52
000376    34    CALL SPAN(LINE,DIGITS,KA,KB,1)                          ENOUGH  53
000403          IF(FAIL.EQ.1)GO TO 25                                   ENOUGH  54
```

```
000413          CALL COPY(LAB,KA,KB)
000436          TABS(M,1)=DECODE(LAB)
000427    25    CALL LOCA(LAB,7,MARK)
000516          CALL STRING(MARK,GOTO,KA,KB,0)
000621          IF(FAIL.EQ.1)GO TO 27
000624          CALL COPY(LAB,SPACE(KB+1),0)
000630          CALL LAST(LAB)
000432          TABS(M,3)=DECODE(LAB)
000436          IF(SPACE(MARK).EQ.3(9))TABS(M,4)=1
000444          GO TO 29
000445    27    CALL INPUT(MARK,RETURN)
000447          IF(FAIL.EQ.0)GO TO 28
000451          CALL IDENT(MARK,STOP)
000453          IF(FAIL.EQ.0)GO TO 28
000455          CALL INPUT(MARK,END)
000457          IF(FAIL.EQ.1)GO TO 26
000462          TABS(M,5)=1
000464          IF(TABS(M,2).EQ.1)GO TO 26
000466          GO TO 15
000466    28    TABS(M,5)=1
000471          IF(TABS(M,2).EQ.1)GO TO 29
000472          FLAG=1
000473          WRITE(6,29)
000477    29    FORMAT(35H **ERROR**THERE MUST BE AN ASSERTION  BEFORE EVERY RETUR
               1N AND STOP**                                                    )
000477    2     I=I+1
000501          IF(ISTACK .EQ. 0 .OR. JFLAG.EQ.1)GO TO 77
000517          CALL ALIST(LOOPI,J(42))
000511          IPX=ISTACK
000512          LOOPJ=0
000514    888   CALL COPY(LOOPI,SPACE(IPX),0)
000520          CALL LAST(LOOPJ)
000522          IPX12=IPX+12
000525          CALL CONCAT(LOOPI,LOOPJ,IPX12)
000529          IPX=SPACE(IPX+1)
000532          IF(IPX.EQ.0)GO TO 888
000521          SPACE(LOOPI)=J(43)
000533          GO TO 2
000534    999   CALL ATLIST(LOOPJ)
000536          IPX(M-1,2)=LOOPI
000540    77    IF(M.EQ.100.OR.IPOINT.EQ.0)GO TO 19
000541          WRITE(6,30)
000554    30    FORMAT(50H **PROGRAM TOO LONG, THE NUMBER OF NON-COMMENT AND NON-A
               1SSERTION CARDS EXCEED 100**                                     )
000555          STOP
000556    16    M=M-1
000560          TABS(M,5)=1
000563          IF(TABS(M,2).EQ.1)GO TO 31
000564          WRITE(6,32)
000567    32    FORMAT(40H **ERROR**THERE MUST BE AN ASSERTION BEFORE THE LAST STA
               1TEMENT OF THE PROGRAM**                                         )
000567          GO TO 31
000571    26    TABS(M,5)=1
000573    31    DO 20 I=1,I
```

```
ENOUGH    55
ENOUGH    56
ENOUGH    57
ENOUGH    58
ENOUGH    59
ENOUGH    60
ENOUGH    61
ENOUGH    62
ENOUGH    63
CORE      124
ENOUGH    64
ENOUGH    65
ENOUGH    66
ENOUGH    67
ENOUGH    68
ENOUGH    69
ENOUGH    70
ENOUGH    71
ENOUGH    72
ENOUGH    73
ENOUGH    74
ENOUGH    75
ENOUGH    76
ENOUGH    77
ENOUGH    78
ENOUGH    79
CORE      125
CORE      16
CORE      17
CORE      18
CORE      19
CORE      20
CORE      21
CORE      22
CORE      23
CORE      24
CORE      25
CORE      26
CORE      27
CORE      127
CORE      128
ENOUGH    81
ENOUGH    82
ENOUGH    83
ENOUGH    84
ENOUGH    85
ENOUGH    86
ENOUGH    87
ENOUGH    88
ENOUGH    89
ENOUGH    90
ENOUGH    91
ENOUGH    92
ENOUGH    93
```

```
000575          IF(TABS(I,3).EQ.0)GO TO 20                                  ENOUGH    94
000576          DO 21 J=1,I                                                 ENOUGH    95
000603          IF(TABS(I,3).EQ.TABS(J,1))GO TO 22                          ENOUGH    96
000603    21    CONTINUE                                                    ENOUGH    97
000605          FLAG=1                                                      ENOUGH    98
000606          WRITE(6,33)TABS(I,3)                                        ENOUGH    99
000614          GO TO 20                                                    CORRC     28
000616    33    FORMAT(39H ***ERROR**THERE IS A TRANSFER TO LABEL ,I6, 19H WHICH I  ENOUGH   100
                IS UNDEFINED)                                             ENOUGH    101
000616    22    TABS(I,3)=J                                                 ENOUGH    102
000620          IREC(I,1)=J                                                 CORRC     129
000621    20    CONTINUE                                                    ENOUGH    103
000624          CALL DELETB(PETION)                                         ENOUGH    104
000625          CALL DELETB(END)                                            ENOUGH    105
000627          CALL DELETB(STOP )                                          ENOUGH    106
000631          CALL DELETB(GOTO)                                           ENOUGH    107
000633          CALL DELETB(LINE)                                           ENOUGH    108
000635          CALL DELETB(IAD)                                            ENOUGH    109
000637          CALL DELETB(LOOPL)                                          CORRC     130
000641          CALL DELETB(LOOPE)                                          CORRC     131
000643          CALL DELETB(JS)                                             CORRC     132
000645          CALL NLIST(IFP,0(42))                                       ENOUGH    110
000650          CALL NLIST(LAPAT,0(45))                                     ENOUGH    111
000652          CALL INSERT(LAPAT,0,1)                                      ENOUGH    112
000656          CALL NLIST(LAPAT,0(46))                                     ENOUGH    113
000661          CALL INSERT(LAPAT,0,0)                                      ENOUGH    114
000664          CYCLIC=0                                                    ENOUGH    115
000665          M=1                                                         ENOUGH    116
000666          PATH=0                                                      ENOUGH    117
000667    2     CALL INSERT(PATH,0,0(45))                                   ENOUGH    118
000672          CALL INSERT(PATH,0,M)                                       ENOUGH    119
000675          CALL INSERT(PATH,0,0(46))                                   ENOUGH    120
000700          CALL INSERT(PATH,0,TABS(M,4))                               ENOUGH    121
000704          IF(TABS(M,5).EQ.0)GO TO 1                                   ENOUGH    122
000707          SPACE(LAPAT)=TABS(M,3)                                      ENOUGH    123
000711          CALL STRING(PATH,LAPAT,KA,KB,0)                             ENOUGH    124
000714          A=TABS(1,3)                                                 ENOUGH    125
000715          IF(FAIL)4,4,?                                               ENOUGH    126
000721    1     IF(TABS(M,5).EQ.1)GO TO 3                                   ENOUGH    127
000724          M=M+1                                                       ENOUGH    128
000725          GO TO 2                                                     ENOUGH    129
000725    4     IF(TABS(M,2).EQ.1)GO TO 3                                   ENOUGH    130
000730          KA=PATH                                                     ENOUGH    131
000741          CALL DELETB(CYCLIC)                                         ENOUGH    132
000742    9     KA=SPACE(KA+1)                                              ENOUGH    133
000745          KA=SPACE(KA+1)                                              ENOUGH    134
000746          CALL INSERT(CYCLIC,0,SPACE(KA))                             ENOUGH    135
000742          IF(SPACE(KA).EQ.0)GO TO 7                                   ENOUGH    136
000746          M=SPACE(KA)                                                 ENOUGH    137
000747          IF(TABS(M,2).EQ.1)GO TO 3                                   ENOUGH    138
000751          KA=SPACE(KA+1)                                              ENOUGH    139
000752          KA=SPACE(KA+1)                                              ENOUGH    140
000754          GO TO 9                                                     ENOUGH    141
000754    7     KC=6                                                        ENOUGH    142
```

```
000755          KA=CYCLIC                                                ENOUGH  143
000757          KA=SPACE(KA)                                             ENOUGH  144
000761          CYCLE=ENCODE(TABS(KB,1))                                 ENOUGH  145
000764    11    KA=SPACE(KA+1)                                           ENOUGH  146
000766          IF(KA.EQ.0)GO TO 8                                       ENOUGH  147
000767          KB=SPACE(KA)                                             ENOUGH  148
000771          IF(TABS(KB,1).EQ.0)GO TO 10                              ENOUGH  149
000772          KC=ENCODE(TABS(KB,1))                                    ENOUGH  150
000775          CALL INSERT(KC,0,0(46))                                  ENOUGH  151
001000          CALL CONCAT(CYCLE,CYCLE,KC)                              ENOUGH  152
001003          CALL DELETE(KC)                                          ENOUGH  153
001005          GO TO 11                                                 ENOUGH  154
001007    10    N=0                                                      ENOUGH  155
001010    14    IF(TABS(KB-N,1).NE.0)GO TO 13                            ENOUGH  156
001013          N=N+1                                                    ENOUGH  157
001014          IF(KB.NE.N)GO TO 14                                      ENOUGH  158
001015          KC=ENCODE(0)                                             ENOUGH  159
001020    12    CALL INSERT(KC,0,0(46))                                  ENOUGH  160
001023          CALL CONCAT(CYCLE,CYCLE,KC)                              ENOUGH  161
001026          CALL DELETE(KC)                                          ENOUGH  162
001030          KC=ENCODE(1)                                             ENOUGH  163
001033          CALL INSERT(KC,0,0(37))                                  ENOUGH  164
001036          CALL CONCAT(CYCLE,CYCLE,KC)                              ENOUGH  165
001041          CALL DELETE(KC)                                          ENOUGH  166
001043          GO TO 11                                                 ENOUGH  167
001045    13    KC=ENCODE(TABS(KB-N,1))                                  ENOUGH  168
001051          GO TO 12                                                 ENOUGH  169
001052    *     CALL CONCAT(CYCLE,CYCLE,IKP)                             ENOUGH  170
001055          CALL INSERT(CYCLE,0,0(41))                               ENOUGH  171
001060          WRITE(6,15)                                              ENOUGH  172
001064    15    FORMAT(' ***ERROR-THE FOLLOWING CYCLE  DOES NOT HAVE AN ASSERTION  ENOUGH  173
                1 ASSOCIATED WITH IT***                                  )ENOUGH  174
001076          CALL DUMP(CYCLE,10,100)                                  ENOUGH  175
001107          CALL DELETE(CYCLE)                                       ENOUGH  176
001071          FLAG=1                                                   ENOUGH  177
001072    *     CALL STACK(PATH,REPAT,KA,KB,0)                           ENOUGH  178
001077          IF(FAIL.EQ.1)GO TO 5                                     ENOUGH  179
001101          KB=PAT1                                                  ENOUGH  180
001103          CALL DELETE(PATH,KB,KA)                                  ENOUGH  181
001106          CALL INSERT(PATH,0,0)                                    ENOUGH  182
001110          CALL FIND(PATH,3,KA)                                     ENOUGH  183
001113          KA=SPACE(KA)+1                                           ENOUGH  184
001116          IF(TABS(I-1,5).EQ.1)GO TO 3                              ENOUGH  185
001120          GO TO 2                                                  ENOUGH  186
001121    5     FAIL=FLAG                                                ENOUGH  187
001123          CALL DELETE(PATH)                                        ENOUGH  188
001126          CALL DELETE(REPAT)                                       ENOUGH  189
001131          CALL DELETE(TEMPAT)                                      ENOUGH  190
001134          CALL DELETE(IKP)                                         ENOUGH  191
001137          CALL DELETE(CYCLIC)                                      ENOUGH  192
001142          IF(FAIL.EQ.1)GO TO 49                                    CORBC    29
001147          DO 81 I=1,100                                            CORB    133
001151          IF(TABS(I,2).EQ.0)GO TO 81                               CORB    134
001162          IF(FLOC(I,1).EQ.0)GO TO 82                               CORB    135
```

```
001143          ISAVE(I,1)=-ISAVE(I,1)                                  COR8    136
001145          GO TO 81                                                COR8    137
001146    82    IREC(I,1)=-1000000                                      COR8    138
001147    81    CONTINUE                                                COR8    139
001151    83    IF(ISTACK.NE.0)IFLAG=1                                  CORBC    30
001153          ISAVE(I,1)=IFLAG                                        COR8    141
001155          IF(IFLAG.EQ.1)GO TO 890                                 CORBC    31
001156                                                                  CORBC    32
001157          DO 876 I=1,100                                          CORBC    33
001160          IF(IREC(I,2).NE.0)RETURN                                CORBC    34
001162    876   CONTINUE                                                CORBC    35
001164          ISAVE(I,1)=1                                            CORBC    36
001165          RETURN                                                  CORBC    37
001166    890   WRITE(6,78)                                             COR8    144
001172    78    FORMAT(////40H IMPROPERLY NESTED LOOP DECLARATIONS***    //      COR8    145
              1***ANALYSIS OF TERMINATION WILL NOT BE COMPLETED**       ///     COR8    146
001172          DO 79 I=1,100                                           COR8    147
001175          IF(IREC(I,2).GT.0)CALL DELETB(IREC(I,2))                COR8    148
001201    79    CONTINUE                                                COR8    149
001203    80    IF(ISTACK.EQ.0)RETURN                                   COR8    150
001205          CALL DELETB(SPACE(ISTACK))                              COR8    151
001211          CALL DELETA(ISTACK,0)                                   COR8    152
001212          GO TO 80                                                ENOUGH  194
001214          END
```

```
                   INTEGER FUNCTION VERCON(PATH,ICE,NPLOOP)                       CORBP      13
000005             COMMON/BLOCK/SPACE(14000),FAIL,FREE                            BIGGER      5
000005             COMMON/DIGIP/IDIG,IOP,ICOM,ISEMI,ICONE/ALPHNU/Q                VERCON      4
                  1/UPPER/ISTK(250,2),IPAD(200)/JOPS/IOPS,IDA,IDS,ILET            CORB      154
000006             INTEGER PATH,Q(63),SPACE,FAIL,FREE,RIGHT,SIMPLI                CORB      155
000006             COMMON/KTAB/ITERM(25,5)                                        CORBP      14
            C THE ENTRY IN THE FIRST COLUMN OF ITERM INDICATES THE PATH NUMBER.   CORBP      15
            C THE ENTRY IN THE SECOND COLUMN INDICATES THE ASSERTION AT WHICH THE PA  CORBP  16
            C STARTS AND THE ENTRY IN THE THIRD COLUMN INDICATES THE ASSERTION AT WH  CORBP  17
            C PATH ENDS.  THE ENTRY IN THE FOUTRH COLUMN IS CODED AS FOLLOWS      CORBP      18
            C      0=ERROR DECREASING LOOP EXPRESSION                             CORBP      19
            C      1=PRIMARY PATH                                                 CORBP      20
            C      2=POSSIBLE PRIMARY PATH                                        CORBP      21
            C      3=SECONDARY PATH                                               CORBP      22
            C IF FOR A CERTAIN ROW, THE ENTRY IN COLUMN 4 IS 2, THEN IN THE SAME ROW  CORBP  23
            C ENTRY IN THE FIFTH COLUMN IS A POINTER TO THE STATEMENT OF CONDITION TH  CORBP 24
            C MAKE THAT PATH A PRIMARY PATH.                                      CORBP      25
            C NPLOOP IS THE INDEX OF THE LAST ROW OF ITERM                        CORBP      26
000006             ILET=0                                                         VERCON      7
000007             DO 9 I=1,25                                                    VERCON      8
000010             CALL INSERT(ILET,0,Q(I))                                       VERCON      9
000017      9      CONTINUE                                                       VERCON     10
000017             IFLAG=0                                                        CORBR       1
000020             NSAVE=0                                                        CORBR       2
000021             INSIN=0                                                        VERCON     11
000022             LINE=0                                                         VERCON     12
000023             LEFT=0                                                         VERCON     13
000024             RIGHT=0                                                        VERCON     14
000025             IHYP=0                                                         CORBP      27
000026             IND=0                                                          CORBP      28
000027             CALL NLIST(IOPS,Q(44))                                         VERCON     15
000031             CALL INSERT(IOPS,0,Q(46))                                      VERCON     16
000034             CALL INSERT(IOPS,0,Q(58))                                      VERCON     17
000037             CALL INSERT(IOPS,0,Q(59))                                      VERCON     18
000042             CALL INSERT(IOPS,0,Q(60))                                      VERCON     19
000045             CALL INSERT(IOPS,0,Q(61))                                      VERCON     20
000050             CALL INSERT(IOPS,0,Q(62))                                      VERCON     21
000053             CALL INSERT(IOPS,0,Q(55))                                      VERCON     22
000056             CALL INSERT(IOPS,0,Q(56))                                      VERCON     23
000061             CALL INSERT(IOPS,0,Q(54))                                      VERCON     24
000064             CALL INSERT(IOPS,0,Q(53))                                      VERCON     25
000067             CALL INSERT(IOPS,0,Q(52))                                      VERCON     26
000072             CALL CONCAT(IOPS,IOPS,IOP)                                     VERCON     27
000075             CALL NLIST(IDA,Q(57))                                          VERCON     28
000100             CALL NLIST(IDS,Q(43))                                          VERCON     29
000103             CALL NLIST(ICP,Q(42))                                          VERCON     30
000106             CALL NLIST(ICG,Q(44))                                          VERCON     31
000111             CALL NLIST(LP,Q(41))                                           VERCON     32
000114             DO 1 I=1,250                                                   VERCON     33
000120             ISTK(I,1)=0                                                    VERCON     34
000121      1      ISTK(I,2)=0                                                    VERCON     35
000124             CALL ANY(PATH,ISEMI,KA,0)                                      VERCON     36
000127             CALL LAN(PATH,5,KB)                                            VERCON     37
```

```
000132          IF(KA.EQ.KA)GO TO 7                                     VERCON    38
000136          CALL COPY(VERCON,KB,KA)                                 VERCON    39
000141          GO TO 8                                                 VERCON    40
000144      7   CALL NLIST(VERCON,Q(47))                                VERCON    41
000147          CALL INSERT(VERCON,0,Q(5))                              VERCON    42
000152          CALL INSERT(VERCON,0,Q(21))                             VERCON    43
000155          CALL INSERT(VERCON,0,Q(13))                             VERCON    44
000160          CALL INSERT(VERCON,0,Q(20))                             VERCON    45
000163          CALL INSERT(VERCON,0,Q(47))                             VERCON    46
000166      8   IPT=SPACE(KA+1)                                         VERCON    47
000170          CALL CONCAT(VERCON,VERCON,IRP)                          VERCON    48
000173          CALL INSERT(VERCON,0,Q(41))                             VERCON    49
000177      2   CALL BREAK(IPT,ISEMI,KA)                                VERCON    50
000201          IF(SPACE(IPT).NE.Q(43))GO TO 3                          VERCON    51
000204          IPT=SPACE(KA+1)                                         VERCON    52
000207          IPT=SPACE(IPT+1)                                        VERCON    53
000211          GO TO 2                                                 VERCON    54
000211      3   CALL COPY(LINE,IPT,KA)                                  VERCON    55
000214          IPT=SPACE(KA+1)                                         VERCON    56
000217          IPT=SPACE(IPT+1)                                        VERCON    57
000221          IF(SPACE(LINE).NE.Q(3))GO TO 4                          VERCON    58
000224          CALL LEN(LINE,4,KA)                                     VERCON    59
000227          KB=LIF                                                  VERCON    60
000231          CALL DELETE(LINE,KB,KA)                                 VERCON    61
000233          CALL FIXIT(LINE)                                        VERCON    62
000236          CALL CONCAT(LINE,LINE,IRP)                              VERCON    63
000241          CALL INSERT(LINE,0,Q(6))                                VERCON    64
000243          CALL INSERT(LINE,0,Q(53))                               VERCON    65
000246          CALL COPY(LINE,VERCON,0)                                COR8P     29
000251          CALL CONCAT(VERCON,VERCON,LINE)                         VERCON    66
000254          IF(EF.EQ.0) GO TO 20                                    COR8      156
000257          WRITE(6,21)                                             COR8      157
000262      21  FORMAT(/' THE CONTROLLED EXPRESSION FOR THIS PATH IS ')  COR10     28
000262          CALL SETUP(ICE,5,100)                                   COR8      159
000267          CALL COPY(VERCON,ICE,0)                                 COR8P     30
000272          CALL FIXIT(ICE)                                         COR8      160
000275          LIF=SIZE(ICE)                                           COR8      161
000302          CALL CONCAT(ICE,ICE,IRP)                                COR8A     6
000306          CALL INSERT(ICE,0,Q(41))                                COR8A     7
000311          WRITE(6,10)                                             COR8      162
000315      10  FORMAT(/' THE VALUE OF CONTROLLED EXPRESSION AT THE END OF THE PATH') COR10  29
000315          CALL SETUP(ICE,5,100)                                   COR8      165
              C SET CELL 5 OF RPLOOP TO POINT TO NULL STRING
000321          ITS4(RPLOOP,5)=0                                        COR8P     31
              C GET RID OF FIRST ( IN NOLD AND ICE                      COR8P     32
000324          CALL DELETA(NOLD,0)                                     COR8P     33
000326          CALL DELETA(ICE,0)                                      COR8P     34
              C TAKE FIRST EXPRESSION OFF OF NOLD AND CALL IT NOLD      COR8P     35
000332      32  K1=NOLD                                                COR8P     36
000334      34  CALL ANY(K1,NOLD,KA)                                   COR8P     37
000337          IF(FAIL.EQ.1)GO TO 30                                   COR8P     38
000343          CALL VAL2(NOLD,KA)                                      COR8P     39
000345          IF(FAIL.EQ.0)GO TO 33                                   COR8P     40
000350          K1=SPACE(K1+1)                                          COR8P     41
                                                                        COR8P     42
```

```
000352          GO TO 34                                                COR8P   43
000353    33    CALL DELETE(MOLD)                                       COR8P   44
000355          CALL COPY(MOLD,NOLD,KA)                                 COR8P   45
000360          IPK12=NOLD                                              COR8P   46
000362          CALL DELETE(NOLD,IPK12,KA)                              COR8P   47
        C THE FIRST EXPRESSION OFF OF ICE AND CALL IT NEW               COR8P   48
000364          K1 = ICE                                                COR8P   49
000367    34    CALL ANY(K1,ICOM,KA)                                    COR8P   50
000372          CALL PAL2(ICE,KA)                                       COR8P   51
000376          IF(FAIL.EQ.0)GO TO 35                                   COR8P   52
000401          K1=SPACE(K,+1)                                          COR8P   53
000403          GO TO 34                                                COR8P   54
000404    35    CALL COPY(NEW,ICE,KA)                                   COR8P   55
000406          IPK12=ICE                                               COR8P   56
000411          CALL DELETE(ICE,IPK12,KA)                               COR8P   57
        C IF THE LOOP EXPRESSION HAS NOT CHANGED,THE GO ON TO NEXT LOOP EXPRESSI COR8P  58
        C IN MOLD AND ICE                                               COR8P   59
000413          CALL DENT(MOLD,NEW)                                     COR8P   60
000415          IF(FAIL.EQ.1)GO TO 31                                   COR8P   61
000421          CALL DELETE(MOLD)                                       COR8P   62
000423          CALL DELETE(NEW)                                        COR8P   63
000425          GO TO 32                                                COR8P   64
        C BUT IF IT HAS CHANGED MAKE THE STRING 0.LE.(NEW)-(MOLD) AND SIMPLIFY I COR8P  65
000430    31    IDIFS=0                                                 COR8P   66
000431          CALL CONCAT(IDIFS,MOLD,IPP)                             COR8P   67
000434          CALL INSERT(IDIFS,0,0(+1))                              COR8P   68
000437          CALL INSERT(IDIFS,0,0(9))                               COR8P   69
000442          CALL INSERT(IDIFS,0,0(+2))                              COR8P   70
000445          IPK12=IDIFS                                             COR8P   71
000447          CALL CONCAT(IDIFS,NEW,IPK12)                            COR8P   72
000451          CALL INSERT(IDIFS,0,0(41))                              COR8P   73
000454          CALL DELETE(NEW)                                        COR8P   74
000456          CALL INSERT(IDIFS,0,0(60))                              COR8P   75
000461          CALL INSERT(IDIFS,,0(27))                               COR8Q   1
000464          IDIFS=SIMPLI(IDIFS)                                     COR8P   77
000467          IF(-1)312,313,311                                       COR8P   78
        C IF THE RESULT OF THE SIMPLIFICATION IS NEITHER TRUE OR FALSE THEN THE COR8P  79
        C RELATION 0.LE.(NEW)-MOLD) IS RETURNED SIMPLIFIED.  CREATE THE STRING  COR8P  80
        C IHYP.IMPLIES.0.LE.(NEW)-(MOLD) WHERE IHYP IS THE HYPOTHESIS OF THE    COR8P  81
        C VERIFICATION CONDITION OF THIS PATH AND SIMPLIFY IT.          COR8P   82
000472    312   CALL INSERT(IDIFS,0,0(56))                              COR8P   83
000475          JDIFS=0                                                 COR8P   84
000476          CALL CONCAT(JDIFS,IHYP,IDIFS)                           COR8P   85
000501          JDIFS=SIMPLI(JDIFS)                                     COR8P   86
000504          CALL DELETE(JDIFS)                                      COR8P   87
000505          IF(FAIL)313,313,311                                     COR8P   88
        C IF THE RESULT OF THIS SIMPLIFICATION IS NEITHER TRUE OR FALSE THEN THE COR8P  89
        C RELATION 0.LE.(NEW)-(MOLD) IS CONCATENATED ONTO ITERM(NPLOOP,5)       COR8P   90
000511    313   IF(ITERM(NPLOOP,5).EQ.0)GO TO 314                       COR8P   91
000513          SPACE(IDIFS)=0(56)                                      COR8P   92
000515          CALL CONCAT(ITERM(NPLOOP,5),ITERM(NPLOOP,5),IDIF)       COR8P   93
000520          CALL DELETE(IDIFS)                                      COR8P   94
000522          GO TO 32                                                COR8P   95
000525    314   CALL DELETE(IDIFS,0)                                    COR8P   96
```

```
                ITERM(NPLOOP,5)=IDIFS                                   COR8P   97
                IDIFS=0                                                 COR8P   98
                GO TO 32                                                COR8P   99
      C IF THE RESULT OF THE SIMPLIFICATION IS TRUE THEN THE LOOP EXPRESSION W  COR8P  100
      C DECREASING- GO ON TO NEXT LOOP EXPRESSION                       COR8P   101
      310       CALL DELETE(IDIFS)                                      COR8P   102
                GO TO 32                                                COR8P   103
      C IF THE RESULT OF THE SIMPLIFICAION IS FALSE THEN THE LOOP EXPRESSION W  COR8P  104
      C DECREASING  THIS IS AN ERROR BUT GO ON ANYWAY TO NEXT LOOP EXPRESSION   COR8P  105
      311   WRITE(6,3110)                                               COR8P   106
      3110  FORMAT(1X,'THIS LOOP EXPRESSION IS DECREASING     ')        COR8P   107
            CALL JUMPTO(NOLD,35,70)                                     COR8P   108
            ITERM(NPLOOP,4)=0                                           COR8P   109
            GO TO 310                                                   COR8P   110
      C ALL OF THE LAST LOOP EXPRESSION OF THE CONTROLLED EXPRESSION HAS BEEN   COR8P  111
      C PUSHED-DECREASING.  NOW THE LAST LOOP EXPRESSION IS CHECKED.  IF IT H   COR8P  112
      C STRICTLY INCREASED AND IS BOUNDED,THEN THE PATH IS A PRIMARY PATH, OTH  COR8P  113
      C IT IS A SECONDARY PATH AND MUST BE IN A LOOP WITH SOME PRIMARY PATH.    COR8P  114
      30        CALL INSERT(NOLD,0,0(41))                               COR8P   115
                CALL INSERT(NOLD,0,0(34))                               COR8P   116
                CALL COPY(LICE,ICE,0)                                   COR8Q     2
                CALL LAST(LICE)                                         COR8Q     3
                CALL REPLACE(ICE,ICE,NOLD)                              COR8P   117
                CALL INSERT(ICE,0,0(41))                                COR8P   118
                CALL COPY(LICE,ICE,0)                                   COR8P   119
                CALL INSERT(LICE,0,0(60))                               COR8P   120
                CALL INSERT(LICE,0,0(27))                               COR8Q     4
      C THE SET (0.LE.(ICE)-(NOLD)) IS FORMED AND THEN SIMPLIFIED TO CHECK TO   COR8P  122
      C THE LAST LOOP EXPRESSION DECREASES                              COR8P   123
                LICE=SIMPLIFY(LICE)                                     COR8P   124
                IF(FAIL)305,306,307                                     COR8P   125
      C IF THE RESULT OF THE SIMPLIFICATION IS NEITHER TRUE OR FALSE THE FORM   COR8P  126
      C SET (HYP.IMPLIES.0.LE.(ICE)-(NOLD)) AND SIMPLIFY IT            COR8P   127
      305   CALL COPY(NSAVE,LICE,0)                                     COR8R     3
            CALL INSERT(LICE,0,0(53))                                   COR8R     4
            HYP=LICE                                                    COR8P   129
            CALL REPLACE(LICE,HYP,IPK12)                                COR8P   130
            LICE=SIMPLIFY(LICE)                                         COR8P   131
            CALL DELETE(LICE)                                           COR8P   132
            IF(FAIL)308,306,307                                         COR8R     5
      308   IF(ITERM(NPLOOP,5).EQ.0)GO TO 309                           COR8R     6
            CALL INSERT(NSAVE,0,0(55))                                  COR8R     7
            CALL REPLACE(ITERM(NPLOOP,5),ITERM(NPLOOP,5),NSAVE)         COR8R     8
            GO TO 310                                                   COR8R     9
      309   ITERM(NPLOOP,5)=NSAVE                                       COR8R    10
            NSAVE=0                                                     COR8R    11
            GO TO 310                                                   COR8R    12
      C IF THE RESULT OF THE SIMPLIFICATION IS FALSE THEN THE THE LAST LOOP EX  COR8P  134
      C DECREASES  THIS IS AN ERROR BUT GO ON ANYWAY TO NEXT STEP.      COR8P   135
      307   CALL DELETE(LICE)                                           COR8P   136
            CALL DELETE(NSAVE)                                          COR8R    13
            WRITE(6,3110)                                               COR8P   137
            CALL DELETE(NOLD)                                           COR8P   138
            CALL JUMPTO(NOLD,35,70)                                     COR8P   139
```

```
                      CALL DELETE(JOLD)                                  COR8P   140
                      IF(SIMPLOP,4)=0                                    COR8P   141
                      CALL DELETE(ICE)                                   COR8P   142
                      GO TO 20                                           COR8P   143
                C IF THE RESULT OF THE SIMPLIFICATION IS NOT FALSE, THEN FORM THE STRING   COR8P   144
                C OLD.LT.ICE WHERE ICE IS THE VALUE OF THE LAST LOOP EXPRESSION AT THE END COR8P   145
                C PATH AND THE VALUE OF THE LAST LOOP EXPRESSION AT THE BEGINNING OF THE   COR8P   146
                C AND SIMPLIFY IT TO SEE OF THE LOOP EXPRESSION INCREASES                  COR8P   147
           303  CALL INSERT(ICE,0,Q(58))                                  COR8P   148
                CALL INSERT(ICE,0,Q(27))                                  COR8Q     5
                CALL DELETE(ISAVE)                                        COR8R    14
                CALL DELETE(JOLD)                                         COR8P   150
                ICE=SIMPLI(ICE)                                           COR8P   151
                IF(FAIL)302,400,301                                       COR8P   152
                C IF THE RESULT OF THE SIMPLIFICATION IS NEITHER TRUE OR FALSE THEN FORM   COR8P   153
                C HYP.IMPLIES. OLT.ICE AND SIMPLIFY THAT.                  COR8P   154
           302  CALL INSERT(ICE,0,Q(54))                                  COR8P   155
                JOLD=0                                                    COR8P   156
                CALL CONCAT(JOLD,HYP,ICE)                                 COR8P   157
                JOLD=SIMPLI(JOLD)                                         COR8P   158
                CALL DELETE(JOLD)                                         COR8P   159
                IF(FAIL)303,300,301                                       COR8P   160
                C IF THE RESULT OF THAT SIMPLIFICATION IS STILL NEITHER TRUE OR FALSE TH   COR8P   161
                C CONCATENATE ICE TO ITERM(NPLOOP,5)                       COR8P   162
           303  IF(ITERM(NPLOOP,5).EQ.0)GO TO 304                         COR8P   163
                SPACE(100)=Q(55)                                          COR8P   164
                CALL CONCAT(ITERM(NPLOOP,5),ITERM(NPLOOP,5),ICE)          COR8P   165
                CALL DELETE(ICE)                                          COR8P   166
                IFLAG=1                                                   COR8R    15
                GO TO 20                                                  COR8Q     6
           304  CALL INSERT(ICE,0)                                        COR8P   172
                ITERM(NPLOOP,5)=ICE                                       COR8P   173
                ICE=1                                                     COR8P   174
                IFLAG=2                                                   COR8R    16
                GO TO 300                                                 COR8Q     7
                C IF THE RESULT OF THE SIMPLIFICATION IS A FAILURE THEN THE PATH IS SECO   COR8P   179
           301  IF(SIMPLOP,4)=3                                           COR8P   180
                CALL DELETE(ICE)                                          COR8P   181
                PRINT 3017                                                COR8P   182
          3017  FORMAT(' THIS IS NOT A PRIMARY PATH')                     COR8P   183
                GO TO 20                                                  COR8P   184
                C SIMPLIFICATION WAS SUCCESSFUL                           COR8Q     8
           300  CALL DELETE(ICE)                                          COR8Q     9
                C NOW TEST TO SEE IF THE LAST LOOP EXPRESSION IS BOUNDED  BY A CONSTANT    COR8Q    10
                C AT THE END OF THE PATH                                  COR8Q    11
                C FIRST TEST TO SEE IF THE EXPRESSION IS ITSELF A CONSTANT (AND THUS       COR8Q    12
                C NUMERICAL)                                              COR8Q    13
                C SEE IF THE FIRST CHARACTER IS A NEGATIVE SIGN            COR8Q    14
           2000 IF(SPACE(ILF).EQ.Q(38))GO TO 2001                         COR8Q    15
                C IF IT IS, GO TO 2001 OTHERWISE SEE IF IT IS A POSITIVE CONSTANT,IF SO    COR8Q    16
                C GO TO 3000                                              COR8Q    17
                CALL SPAN(ILF,IDIG,KS,KR,1)                               COR8Q    18
                IF(FAIL.EQ.1)GO TO 2002                                   COR8Q    19
                IF(SPACE(KR+1).EQ.0)GO TO 3000                            COR8Q    20
```

```
 001040         GO TO 2002                                                 COR8Q   21
          C SEE IF IT IS A NEGATIVE CONSTANT, IF SO, GO TO 3000            COR8Q   22
 001040   2001  CALL SPAN(SPACE(ILE+1),IDIG,KS,KR,1)                       COR8Q   23
 001045         IF(FAIL.EQ.1)GO TO 2002                                    COR8Q   24
 001051         IF(SPACE(KR+1).EQ.0)GO TO 3000                             COR8Q   25
          C IF IT IS NEITHER A NEGATIVE CONSTANT OR A POSITIVE CONSTANT, TEST TO   COR8Q   26
          C SEE IF THE HYPOTHESIS OF THE VERIFICATION CONDITION IMPLIES THAT IT    COR8Q   27
          C IS BOUNDED                                                     COR8Q   28
          C FIRST SIMPLIFY THE HYPOTHESIS.                                 COR8Q   29
 001053   2002  IHYP=SIMPLI(IHYP)                                          COR8Q   30
          C THEN CALL SEPRAT TO GET A LIST OF ALL THE CONJUNCTS IN IHYP THAT ARE   COR8Q   31
          C RELATIONS                                                      COR8Q   32
 001055         IF(SPACE(IHYP).EQ.Q(41))GO TO 2003                         COR8Q   33
 001062         CALL CONCAT(IHYP,IHYP,IRP)                                 COR8Q   34
 001064         CALL INSERT(IHYP,0,Q(41))                                  COR8Q   35
 001067   2003  LIST1=0                                                    COR8Q   36
 001070         LIST2=0                                                    COR8Q   37
 001071         CALL NLIST(IAND,Q(55))                                     COR8Q   38
 001075         CALL NLIST(IPATIO,Q(44))                                   COR8Q   39
 001076         CALL INSERT(IPATIO,0,Q(52))                                COR8Q   40
 001101         CALL INSERT(IPATIO,0,Q(58))                                COR8Q   41
 001104         CALL INSERT(IPATIO,0,Q(59))                                COR8Q   42
 001107         CALL INSERT(IPATIO,0,Q(60))                                COR8Q   43
 001112         CALL INSERT(IPATIO,0,Q(61))                                COR8Q   44
 001115         CALL SEPRAT(IHYP,LIST1,LIST2,IAND,IPATIO)                  COR8Q   45
 001121         CALL DELETE(LIST2)                                         COR8Q   46
 001123         CALL DELETE(IAND)                                          COR8Q   47
 001125         CALL DELETE(IPATIO)                                        COR8Q   48
 001127         IF(LIST1.EQ.0)GO TO 4000                                   COR8Q   49
          C NOW FORM THE STRING(0.LT.-LLF),CALL IT IREL AND CALL DIFF WITH IT     COR8Q   50
 001132         CALL COPY(IREL,LLF,0)                                      COR8Q   51
 001135         CALL CONCAT(IREL,IREL,IRP)                                 COR8Q   52
 001140         CALL INSERT(IREL,0,Q(41))                                  COR8Q   53
 001143         CALL INSERT(IREL,0,Q(38))                                  COR8Q   54
 001146         IREL=SIMPLI(IREL)                                          COR8Q   55
 001151         CALL CONCAT(IREL,IREL,IRP)                                 COR8Q   56
 001153         CALL INSERT(IREL,0,Q(53))                                  COR8Q   57
 001156         CALL INSERT(IREL,0,Q(27))                                  COR8Q   58
 001161         CALL INSERT(IREL,0,Q(41))                                  COR8Q   59
 001164         CALL DIFF(IREL,0,0,0)                                      COR8Q   60
 001167   2006  LOOK=SPACE(LIST1)                                          COR8Q   61
 001171         CALL DIFF(IREL,LOOK,ISIGN,KS)                              COR8Q   62
 001174         IF(ISIGN.NE.2)GO TO 2007                                   COR8Q   63
 001200         CALL DELETE(LOOK)                                          COR8Q   64
 001202         CALL DELETA(LIST1,0)                                       COR8Q   65
 001204         IF(LIST1.NE.0)GO TO 2006                                   COR8Q   66
 001211         CALL DELETE(IREL)                                          COR8Q   67
          C IF IT IS BOUNDED,GO TO 2007                                    COR8Q   68
          C OTHERWISE,THE LAST LOOP EXPRESSION IS NOT BOUNDED BY A CONSTANT AT THE   COR8R   17
          C END OF THE PATH AND THUS THE PATH IS NOT PRIMARY               COR8R   18
 001212   4000  CALL DELETE(LLF)                                           COR8R   19
 001214         PRINT 3017                                                 COR8R   20
 001220         IF(IFLAG.EQ.0)GO TO 20                                     COR8R   21
 001223         IF(IFLAG.EQ.2)GO TO 4004                                   COR8R   22
```

```
001225          KK=ITERM(NPLOOP,5)                                      CORBR   23
001227          CALL MLIST(IAND)                                        CORBR   24
001230    4005  CALL ANY(KK,IAND,KS,0)                                  CORBR   25
001233          IF(KALL.EQ.1)GO TO 4006                                 CORBR   26
001237          KSAVE=KS                                                CORBR   27
001241          KK=SPACE(KS+1)                                          CORBR   28
001242          GO TO 4005                                              CORBR   29
001243    4006  CALL DELETE(ITERM(NPLOOP,5),KSAVE,0)                    CORBR   30
001246          CALL DELET3(IAND)                                       CORBR   31
001250          GO TO 20                                                CORBR   32
001253    4004  CALL DELET3(ITERM(NPLOOP,5))                            CORBR   33
001256          GO TO 20                                                CORBR   34
001260    4003  PRINT 4002                                              CORBQ   84
001264    4002  FORMAT(* THIS IS A PRIMARY PATH IF*)                    CORBQ   85
001266          CALL DUMP(ITERM(NPLOOP,5),35,70)                        CORBQ   86
001271          ITERM(NPLOOP,4)=2                                       CORBQ   87
001275          GO TO 2)                                                CORBQ   88
001275    4001  ITERM(NPLOOP,5)=LLE                                     CORBQ   89
001277          LLE=)                                                   CORBQ   90
001300          GO TO 4003                                              CORBQ   91
001303    2007  CALL DELETE(LLE)                                        CORBQ   92
001307          CALL DELET3(IREL)                                       CORBQ   93
001312    3 00  CALL DELET3(LINK)                                       CORBQ   94
001315          CALL DELETE(LIST1,0)                                    CORBQ   95
001320          IF(LIST1.EQ.0)GO TO 3000                                CORBQ   96
001323          LINK=SPACE(LIST1)                                       CORBQ   97
001326          GO TO 2 )P                                              CORBQ   98
                C THE LAST LOOP EXPRESSION IS BOUNDED                   CORBQ   99
001332    3000  IF(ITERM(NPLOOP,3).NE.0)GO TO 4003                      CORBQ  100
001337          PRINT 3001                                              CORBQ  101
001342    3001  FORMAT(* THIS IS A PRIMARY PATH*)                       CORBQ  102
001345          ITERM(NPLOOP,4)  =1                                     CORBQ  103
001350          GO TO 2)                                                CORBQ  104
001353    4     CALL CHR(LLE,6,KA)                                      VERCON  68
001361          KC=SPACE(KA+1)                                          VERCON  69
001364          IF(SPACE(KC).NE.Q(1).AND.SPACE(KC).NE.Q(62))GO TO 5     VERCON  70
001365          KB=LINE                                                 VERCON  71
001366          CALL DELETE(LINE,KB,KA)                                 VERCON  72
001371          CALL FIRST(LINE)                                        VERCON  73
001373          CALL INSERT(LINE,0,0(55))                               VERCON  74
001377          CALL CP CAT(VERCON,VERCON,LINE)                         VERCON  75
001402          GO TO 2                                                 VERCON  76
001403          CALL ANY(KC,IRP,KB,0)                                   VERCON  77
001407          IF(KALL.EQ.1)GO TO 2                                    VERCON  78
001413          CALL DELETE(LIST)                                       VERCON  79
001416          CALL COPY(LIST,KC,KB)                                   VERCON  80
001421          CALL COPY(RIGHT,SPACE(KB+1),0)                          VERCON  81
001425          CALL FIRST(LIST)                                        VERCON  82
001430          CALL ANY(LEFT,LP,KA,0)                                  VERCON  83
001433          IF(KALL.EQ.1)GO TO 6                                    VERCON  84
001436          CALL ANY(LEFT,IRP,KB,0)                                 VERCON  85
001442          CALL DELETE(LISTDE)                                     VERCON  86
001445          CALL COPY(LISTDE,SPACE(KA+1),KB)                        VERCON  87
001451          CALL FIRST(LISTDE)                                      VERCON  88
```

```
                CALL DELETS(LEFT,KA,K6)                                 VERCON   89
                CALL CONCAT(INSIDE,INSIDE,IRP)                          VERCON   90
                CALL INSTR(INSIDE,0,$(41))                              VERCON   91
                CALL CONCAT(LEFT,LEFT,INSIDE)                           VERCON   92
                LEFT=SIMPLE(LEFT)                                       VERCON   93
                INC=0                                                   VERCON   94
                CALL ANY(LEFT,LP,KA,0)                                  VERCON   95
                IF(KA.EQ.0)INC=125                                      VERCON   96
                DO 431 II=1,125                                         COR8P   191
                IF(ISIM(INC+II,1).EQ.0)GO TO 432                        COR8P   192
                CALL INSERT(ISIM(INC+II,1),LEFT)                        VERCON   99
                IF(KA.EQ.0)GO TO 433                                    COR8P   193
431   CONTINUE                                                          COR8P   194
432   CALL COPY(ISIM(INC+II,1),LEFT,0)                                  COR8P   195
433   CALL COPY(ISIM(INC+II,2),RIGHT,0)                                 COR8P   196
                GO TO 2                                                 VERCON  104
23    CALL DELETS(PATH)                                                 VERCON  105
                CALL DELETS(RIGHT)                                      VERCON  106
                CALL DELETS(LEFT)                                       VERCON  107
                CALL DELETS(INSIDE)                                     VERCON  108
                CALL DELETS(LP)                                         VERCON  109
                CALL DELETS(ISP)                                        VERCON  110
                CALL DELETS(TIME)                                       VERCON  111
                CALL DELETS(TEMP)                                       VERCON  112
                CALL DELETS(TEMPS)                                      VERCON  113
                CALL DELETS(TEMA)                                       VERCON  114
                CALL DELETS(TDS)                                        VERCON  115
                CALL DELETS(ILET)                                       VERCON  116
                CALL DELETS(IDYP)                                       COR8P   197
                CALL DELETS(ZERO)                                       COR8P   198
                DO 21 I=1,125                                           VERCON  117
                IF(ISIM(I,1).EQ.0)GO TO 22                              VERCON  118
                CALL DELETS(ISIM(I,1))                                  VERCON  119
                CALL DELETS(ISIM(I,2))                                  VERCON  120
21    CONTINUE                                                          VERCON  121
22    DO 23 I=126,250                                                   VERCON  122
                IF(ISIM(I,1).EQ.0)RETURN                                VERCON  123
                CALL DELETS(ISIM(I,1))                                  VERCON  124
                CALL DELETS(ISIM(I,2))                                  VERCON  125
23    CONTINUE                                                          VERCON  126
                END                                                     VERCON  127
```

```
                SUBROUTINE FIXIT(LINE)                                  FIXIT      2
000003          COMMON/BLOCK/SPACE(14000),FAIL,FREE                     BIGGER     6
000003          COMMON/SPKEEP/ISIM(250,2),IPAD(200)/ALPHNU/Q(63)/       CORR     167
                1IOPS/IOPS,IOA,IOS,ILET/DIGOP/IDIG,I1,I2,I3,I4          FIXIT      5
000003          INTEGER SPACE,FAIL,FREE,SIMPLE          ,Q              FIXIT      6
000003          CALL SPAN(LINE,IDIG,KA,KB,1)                            FIXIT      7
000006          IF(FAIL.EQ.0.AND.SPACE(KB+1).EQ.0)RETURN                FIXIT      8
000015          CALL GLIST(ISEG,Q(37))                                  FIXIT      9
000021          CALL CONCAT(LINE,LINE,ISEG)                             FIXIT     10
000025          CALL DELETE(ISEG)                                       FIXIT     11
000027          LD=0                                                    FIXIT     12
000030          CALL CONCAT(LD,ILET,IDIG)                               FIXIT     13
000033          CALL GLIST(LD,Q(42))                                    FIXIT     14
000036          CALL INSERT(LINE,0,Q(37))                               FIXIT     15
000042          CALL GLIST(J,0)                                         FIXIT     16
000044    2     CALL SPAN(LINE,IOPS,KA,KB,0)                            FIXIT     17
000050          CALL COPY(ISEG,LINE,KB)                                 FIXIT     18
000054          CALL CONCAT(J,J,ISEG)                                   FIXIT     19
000057          KA=LINE                                                 FIXIT     20
000061          CALL DELETE(LINE,KA,KB)                                 FIXIT     21
000063          IF(LINE.EQ.0)GO TO 1                                    FIXIT     22
000065          CALL ANY(LINE,ILET,KA,1)                                FIXIT     23
000067          IF(FAIL.EQ.1)GO TO 2                                    FIXIT     24
000072          CALL SPAN(LINE,LD,KA,KB,1)                              FIXIT     25
000075          CALL COPY(ISEG,KA,KB)                                   FIXIT     26
000100          CALL DELETE(LINE,KA,KB)                                 FIXIT     27
000103          DO 3 I=1,125                                            FIXIT     28
000106          IF(ISIM(I,1).E).0)GO TO 4                               FIXIT     29
000107          CALL INCUT(ISIM(I,1),ISEG)                              FIXIT     30
000112          IF(FAIL.EQ.0)GO TO 5                                    FIXIT     31
000114    3     CONTINUE                                                FIXIT     32
000116    4     CALL CONCAT(J,J,ISEG)                                   FIXIT     33
000121          GO TO 2                                                 FIXIT     34
000123    5     CALL COPY(ISEG,ISIM(I,2),0)                             FIXIT     35
000127          CALL CONCAT(ISEG,ISEG,IOS)                              FIXIT     36
000132          CALL INSERT(ISEG,0,Q(57))                               FIXIT     37
000135          GO TO 4                                                 FIXIT     38
000137    1     CALL DELETA(J,0)                                        FIXIT     39
000141          IF(ISIM(12,1).EQ.0)GO TO 16                             FIXIT     40
000143          CALL ANY(J,IS,KA,0)                                     FIXIT     41
000146          IF(FAIL.EQ.1)GO TO 16                                   FIXIT     42
000151          CALL GLIST(LINE,0)                                      FIXIT     43
000153    9     CALL SPAN(J,IOPS,KA,KB,0)                               FIXIT     44
000157    8     CALL COPY(ISEG,J,KB)                                    FIXIT     45
000162          CALL CONCAT(LINE,LINE,ISEG)                             FIXIT     46
000165          KB=J                                                    FIXIT     47
000170          CALL DELETE(J,KA,KB)                                    FIXIT     48
000172          IF(J.EQ.0)GO TO 6                                       FIXIT     49
000174          IF(SPACE(J).NE.0(57))GO TO 7                            FIXIT     50
000176          CALL ANY(J,IOS,KB,0)                                    FIXIT     51
000201          SPACE(KB)=        0(42)                                 FIXIT     52
000203          SPACE(J)=0(41)                                          FIXIT     53
000206          GO TO 8                                                 FIXIT     54
```

```
000207   7      CALL ANY(J,TLET,KA,1)                                  FIXIT   55
000212          IF(FAIL.EQ.1)GO TO 9                                   FIXIT   56
000215          CALL SPACE(J,LO),KA,KC,1)                              FIXIT   57
000221          KA=SPACE(KC+1)                                         FIXIT   58
000223          IF(SPACE(KA).NE.O(41))GO TO 9                          FIXIT   59
000226          CALL ANY(KA,IP,KB,0)                                   FIXIT   60
000231          CALL PAL?(SPACE(KA+1),KB)                              FIXIT   61
000234          IF(FAIL.EQ.1)GO TO 9                                   FIXIT   62
000237          CALL COPY(ISEG,KA,KB)                                  FIXIT   63
000242   11     CALL ANY(ISEG,IDS,KT,0)                                FIXIT   64
000245          IF(FAIL.EQ.1)GO TO 10                                  FIXIT   65
000250          SPACE(KT)=O(41)                                        FIXIT   66
000252          GO TO 11                                               FIXIT   67
000253   10     CALL ANY(ISEG,IDS,KT,0)                                FIXIT   68
000256          IF(FAIL.EQ.1)GO TO 12                                  FIXIT   69
000261          SPACE(KT)=O(42)                                        FIXIT   70
000263          GO TO 10                                               FIXIT   71
000264   12     CALL INSERT(ISEG,0,Q(1))                               FIXIT   72
000267          M=SIMPLI(ISEG)                                         FIXIT   73
000272          CALL DELETA(M,0)                                       FIXIT   74
000274          CALL COPY(ISEG,J,KC)                                   FIXIT   75
000277          KA=J                                                   FIXIT   76
000301          CALL DELETC(J,KA,K9)                                   FIXIT   77
000303          CALL CONCAT(ISEG,ISEG,M)                               FIXIT   78
000306          CALL DELETA(M)                                         FIXIT   79
000310          DO 13 I=125,250                                        FIXIT   80
000313          IF(ISIM(I,1).EQ.J)GO TO 14                             FIXIT   81
000314          CALL INSRT(ISIM(I,1),ISEG)                             FIXIT   82
000317          IF(FAIL.EQ.J)GO TO 15                                  FIXIT   83
000321   13     CONTINUE                                               FIXIT   84
000323   14     CALL CONCAT(LINE,LINE,ISEG)                            FIXIT   85
000326          GO TO 8                                                FIXIT   86
000330   15     CALL COPY(ISEG,ISIM(I,2),0)                            FIXIT   87
000334          GO TO 14                                               FIXIT   88
000336   6      CALL DELETA(LINE,0)                                    FIXIT   89
000340   17     CALL DELETA(LINE,0)                                    FIXIT   90
000342          CALL LAST(LINE)                                        FIXIT   91
000345          CALL DELETA(ISEG)                                      FIXIT   92
000347          CALL DELETA(LO)                                        FIXIT   93
000351          CALL DELETA(IR)                                        FIXIT   94
000353          RETURN                                                 FIXIT   95
000354   16     LINE=J                                                 FIXIT   96
000355          J=0                                                    FIXIT   97
000357   18     CALL ANY(LINE,IDS,KA,J)                                FIXIT   98
000362          IF(FAIL.EQ.1)GO TO 19                                  FIXIT   99
000365          SPACE(KA)=O(41)                                        FIXIT  100
000367          GO TO 18                                               FIXIT  101
000373   19     CALL ANY(LINE,IDS,KA,J)                                FIXIT  102
000375          IF(FAIL.EQ.1)GO TO 17                                  FIXIT  103
000376          SPACE(KA)=O(42)                                        FIXIT  104
000400          GO TO 19                                               FIXIT  105
000401          END                                                    FIXIT  106
```

```
                SUBROUTINE RIDB00(PATH)                                  RIDB00    2
000003          COMMON/ALPHA/Q(63)/BLOCK/SPACE(14000),FAIL,IFQQ          BIGGER    7
000003          INTEGER SPACE,FAIL,PATX,PATH,Q                           RIDB00    4
000003          DATA JFALSE/2HFA/,JTRUE/2HTR/                            COR6      1
000003          CALL MLIST(PATX,Q(47))                                   RIDB00    5
000006          CALL INSERT(PATX,0,Q( 5))                                RIDB00    6
000011          J1=PATX                                                  RIDB00    7
000013          CALL INSERT(PATX,0,Q(12))                                RIDB00    8
000016          J2=PATX                                                  RIDB00    9
000021          CALL INSERT(PATX,0,Q(47))                                RIDB00   10
000025          J3 = PATX                                                RIDB00   11
000025          CALL MOBLT1(PATH,PATX,Q(60))                             RIDB00   12
000030          SPACE(J1)=J(20)                                          RIDB00   13
000032          CALL MOBLT1(PATH,PATX,Q(58))                             RIDB00   14
000036          SPACE(J2)=J( 7)                                          RIDB00   15
000041          CALL MOBLT1(PATH,PATX,Q(59))                             RIDB00   16
000044          SPACE(J1)=J( 5)                                          RIDB00   17
000045          CALL MOBLT1(PATH,PATX,Q(61))                             RIDB00   18
000052          SPACE(J2)=J(14)                                          RIDB00   19
000054          CALL MOBLT1(PATH,PATX,Q(52))                             RIDB00   20
000061          SPACE(J2)=J( 5)                                          RIDB00   21
000062          SPACE(J1)=J(17)                                          RIDB00   22
000064          CALL MOBLT1(PATH,PATX,Q(44))                             RIDB00   23
000071          SPACE(J2)=J(15)                                          RIDB00   24
000072          SPACE(J1)=J(13)                                          RIDB00   25
000074          CALL MOBLT1(PATH,PATX,Q(54))                             RIDB00   26
000101          SPACE(J1)=J(20)                                          RIDB00   27
000102          SPACE(J2)=J(14)                                          RIDB00   28
000105          CALL INSERT(PATX,0,Q(47))                                RIDB00   29
000107          CALL MOBLT1(PATH,PATX,Q(62))                             RIDB00   30
000113          SPACE(J3)=J( 1)                                          RIDB00   31
000115          SPACE(J2)=J(14)                                          RIDB00   32
000117          SPACE(J1)=J( 4)                                          RIDB00   33
000121          CALL MOBLT1(PATH,PATX,Q(55))                             RIDB00   34
000125          SPACE(J1)=J(19)                                          RIDB00   35
000127          SPACE(J2)=J(5)                                           RIDB00   36
000131          SPACE(J3)=J(7)                                           RIDB00   37
000133          SPACE(PATX)=J(12)                                        RIDB00   38
000135          CALL INSERT(PATX,0,Q(16))                                RIDB00   39
000140          CALL INSERT(PATX,0,Q(13))                                RIDB00   40
000143          CALL INSERT(PATX,0,Q(9))                                 RIDB00   41
000146          CALL INSERT(PATX,0,Q(47))                                RIDB00   42
000151          CALL MOBLT1(PATH,PATX,Q(53))                             RIDB00   43
000155          CALL DELETA(PATX,0)                                      COR6      2
000157          CALL DELETA(PATX,0)                                      COR6      3
000161          J5 = PATX                                                COR6      4
000162          SPACE(J5)=J(47)                                          COR6      5
000165          J6=SPACE(J5+1)                                           COR6      6
000166          SPACE(J5)=J(5)                                           COR6      7
000173          J4=SPACE(J5+1)                                           COR6      8
000171          SPACE(J4)=J(1)                                           COR6      9
000173          SPACE(J5)=J(12)                                          COR6     10
000175          SPACE(J2)=J(15)                                          COR6     11
```

```
000177          SPACE(J1)=J(6)                                          COR6    12
000201          CALL ROLL(PATH,PATX,JFALSE)                             COR6    13
000204          CALL DELETE(PATX,0)                                     COR6    14
000206          SPACE(J5)=J(47)                                         COR6    15
000210          SPACE(J4)=J(20)                                         COR6    16
000212          SPACE(J3)=J(14)                                         COR6    17
000214          SPACE(J2)=J(21)                                         COR6    18
000216          CALL ROLL(PATH,PATX,JTRUE)                              COR6    19
000221          CALL DELETE(PATX)                                       RIDB00  44
000223          CALL BLIST(PATX,0(39))                                  RIDB00  45
000226          CALL INSERT(PATX,0,0(39))                               RIDB00  46
000231          CALL ROLL(PATH,PATX,0(56))                              RIDB00  47
000235          CALL DELETE(PATX)                                       RIDB00  48
000237          RETURN                                                  RIDB00  49
000240          END                                                     RIDB00  50
```

```
            SUBROUTINE [illegible](PATH,PATX,JSYM)                          BOOLDO     2
[illegible]    COMMON/ [illegible]/BLOCK/SPACE(14000),FAIL,IFQQ              BIGGER     8
[illegible]    INTEGER PATH,PATX,SPACE,FAIL,Q                                BOOLDO     4
[illegible] 1  CALL [illegible](PATH,PATX,K,KK,0)                            BOOLDO     5
[illegible]    IF(FAIL.[illegible]) RETURN                                   BOOLDO     6
[illegible]    SPACE([illegible])=JSYM                                       BOOLDO     7
[illegible]    IPK12=SPACE(K+1)                                              BOOLDO     8
[illegible]    CALL DELETE(PATH,IPK12,KK)                                    BOOLDO     9
[illegible]    GO TO 1                                                       BOOLDO    10
[illegible]    END                                                           BOOLDO    11
```

208

```
                      INTEGER FUNCTION COMPRE(PROG)                                  COMPRE     2
                      INTEGER SPACE,FAIL,FREE,PROG,Q,DIGITS,G                        COMPRE     3
                      COMMON/BLOCK/SPACE(14000),FAIL,FREE/ALPHNU/Q(63)/DIGOP/DIGITS,II,  BIGGER   9
                     1 I2,I3,I4/GOO/IGOO                                             COMPRE     5
                      IFLAG=0                                                        COMPRE     6
                      KL1=0                                                          COMPRE     7
                      CALL ALIST(ILN,Q(63))                                          COMPRE     8
                      CALL INSERT(ILN,0,Q(4))                                        COMPRE     9
                      CALL INSERT(IPS,0,Q(14))                                       COMPRE    10
                      CALL INSERT(IPS,0,Q(5))                                        COMPRE    11
                      CALL ALIST(IRET,Q(63))                                         COMPRE    12
                      CALL INSERT(IRET,0,Q(14))                                      COMPRE    13
                      CALL INSERT(IRET,0,Q(18))                                      COMPRE    14
                      CALL INSERT(IRET,0,Q(21))                                      COMPRE    15
                      CALL INSERT(IRET,0,Q(20))                                      COMPRE    16
                      CALL INSERT(IRET,0,Q(6))                                       COMPRE    17
                      CALL INSERT(IRET,0,Q(18))                                      COMPRE    18
                      CALL ALIST(ISTP,Q(63))                                         COMPRE    19
                      CALL INSERT(ISTP,0,Q(16))                                      COMPRE    20
                      CALL INSERT(ISTP,0,Q(15))                                      COMPRE    21
                      CALL INSERT(ISTP,0,Q(20))                                      COMPRE    22
                      CALL INSERT(ISTP,0,Q(17))                                      COMPRE    23
                      CALL ALIST(IPL,Q(41))                                          COMPRE    24
                      CALL INSERT(IPL,0,Q(6))                                        COMPRE    25
                      CALL INSERT(IPL,0,Q(9))                                        COMPRE    26
                      CALL ALIST(IBG,(44))                                           COMPRE    27
                      LA=0                                                           COMPRE    28
                      LB=0                                                           COMPRE    29
                      C=(1)                                                          COMPRE    30
                      II=0                                                           COMPRE    31
                      CALL ALIST(LSTC,Q(15))                                         COMPRE    32
                      CALL INSERT(SOLN,0,Q(20))                                      COMPRE    33
                      CALL INSERT(SOLD,0,Q(15))                                      COMPRE    34
                      CALL INSERT(SOLH,0,Q(7))                                       COMPRE    35
                      CALL ALIST(LIL,Q(45))                                          COMPRE    36
                1     CALL COPY(PROG,I3,KA,0)                                        COMPRE    37
                      IF(FAIL.EQ.1)GO TO 8                                           COMPRE    38
                      CALL COPY(LINE,PROG,KA)                                        COMPRE    39
                      IPK12=PROG                                                     COMPRE    40
                      CALL DELETE(PROG,IPK12,KA)                                     COMPRE    41
                      CALL SCAN(LINE,(COMD),KA,KB,I)                                 COMPRE    42
                      IF(FAIL.EQ.1)GO TO 9                                           COMPRE    43
                10    CALL SPAN(LINE,IBL,KA,KB,J)                                    COMPRE    44
                      IF(FAIL.EQ.1)GO TO 2                                           COMPRE    45
                      CALL DELETE(LINE,KA,KB)                                        COMPRE    46
                      GO TO 10                                                       COMPRE    47
                2     IF(SPACE(LIL).EQ.J(3))GO TO2                                   COMPRE    48
                      CALL LEN(LINE,A,KA)                                            COMPRE    49
                      IF(SPACE(KA).EQ.Q(45).AND.IFLAG.EQ.1)GO TO 13                  COMPRE    50
                      CALL COPY(L30,LINE,KA)                                         COMPRE    51
                      IPK12=LINE                                                     COMPRE    52
                      CALL DELETE(LINE,IPK12,KA)                                     COMPRE    53
                      CALL COPY(LINE,LINE,0)                                         COMPRE    54
```

```
000212          CALL SPAN(LAB,DIGITS,KA,KB,0)                                   COMPRE  55
000216          IF(FAIL.EQ.1)GO TO 3                                            COMPRE  56
000221          CALL COPY(LAB,KA,KB)                                            COMPRE  57
000224          N=ISIZE(LAB)                                                    COMPRE  58
000226          N=6-N                                                           COMPRE  59
000231          DO 4 I=1,N                                                      COMPRE  60
000233          CALL INSERT (LAB,N,Q(45))                                       COMPRE  61
000236    4     CONTINUE                                                        COMPRE  62
000242          CALL COPY(LAB,LAB,,0)                                           COMPRE  63
000244    3     CALL SPAN(LINE,IN,KA,KB,0)                                      COMPRE  64
000251          IF(FAIL.EQ.1)GO TO 5                                            COMPRE  65
000253          CALL DELETE(LINE,KA,KB)                                         COMPRE  66
000256          GO TO 2                                                         COMPRE  67
000260    5     CALL STRING(LINE,GOTO,KA,KB,1)                                  COMPRE  68
000266          IF(FAIL.EQ.1)GO TO 6                                            COMPRE  69
000267          CALL SPAN(SPACE(K+1),DIGITS,KA,KC,1)                            COMPRE  70
000274          IF(FAIL.EQ.1)GO TO 7                                            COMPRE  71
000277          KC=SPACE(KC+1)                                                  COMPRE  72
000301          IF(SPACE(KC).NE.Q(63))GO TO 7                                   COMPRE  73
000303          CALL INSERT(LINE,4,Q(45))                                       COMPRE  74
000306          CALL INSERT(LINE,2,Q(45))                                       COMPRE  75
000311          GO TO 11                                                        COMPRE  76
000313    6     CALL INSERT (LINE,0,Q(42))                                      COMPRE  77
000316          CALL STRING(LINE,GOTO,KA,KC,0)                                  COMPRE  78
000322          CALL DELETA(GOTO,0)                                             COMPRE  79
000324          IF(FAIL.EQ.1)GO TO 7                                            COMPRE  80
000327          CALL SPAN(SPACE(KC+1),DIGITS,KB,K,1)                            COMPRE  81
000334          IF(FAIL.EQ.1)GO TO 7                                            COMPRE  82
000337          K=SPACE(K+1)                                                    COMPRE  83
000341          IF(SPACE(K).NE.Q(-3))GO TO 7                                    COMPRE  84
000343          CALL INSERT(KA,F,Q(45))                                         COMPRE  85
000346          CALL INSERT(KA,3,Q(45))                                         COMPRE  86
000351          GO TO 11                                                        COMPRE  87
000353    7     CALL INSERT(LINE,IRET)                                          COMPRE  88
000356          IF(FAIL.EQ.0)GO TO 11                                           COMPRE  89
000357          CALL INSERT(LINE,ISTP)                                          COMPRE  90
000361          IF(FAIL.EQ.0)GO TO 11                                           COMPRE  91
000363          CALL STRING(LINE,IFL,KA,KB,1)                                   COMPRE  92
000367          IF(FAIL.EQ.0)GO TO 11                                           COMPRE  93
000371          CALL INSERT(LINE,ICN)                                           COMPRE  94
000373          IF(FAIL.EQ.0)GO TO 11                                           COMPRE  95
000375          CALL SUB(LINE,1,KA,0)                                           COMPRE  96
000402          IF(FAIL.EQ.1)GO TO 12                                           COMPRE  97
000403          CALL SUB2(LINE,KA)                                              COMPRE  98
000408          IF(FAIL.EQ.1)GO TO 12                                           COMPRE  99
000410    11    IPK12=LINE                                                      COMPRE 100
000411          ILAG=0                                                          COMPRE 101
000413          CALL CONCAT(LINE,LAB,IPK12)                                     COMPRE 102
000415    2     IPK1=COMPRE                                                     COMPRE 103
000417          CALL CONCAT(COMPRE,IPK12,LINE)                                  COMPRE 104
000421          GO TO 1                                                         COMPRE 105
000422    12    CALL CONCAT(LINE,LAB,KLINE)                                     COMPRE 106
000426          ILAG=1                                                          COMPRE 107
000427    1     CALL INSERT(LINE,0,Q(43))                                       COMPRE 108
```

```
00432          GO TO 2                                          COMPRE  109
00436     8    CALL DELETA(GOTO)                                COMPRE  110
00436          CALL DELETA(LINE)                                COMPRE  111
00440          CALL DELETA(LAG)                                 COMPRE  112
00442          CALL DELETA(LAG1)                                COMPRE  113
00444          CALL DELETA(ISL)                                 COMPRE  114
00446          CALL DELETA(IFM)                                 COMPRE  115
00450          CALL DELETA(KLINE)                               COMPRE  116
00452          CALL DELETA(IRET)                                COMPRE  117
00456          CALL DELETA(ISTP)                                COMPRE  118
00464          CALL DELETA(ITFL)                                COMPRE  119
00470          CALL DELETA(IFO)                                 COMPRE  120
00502          RETURN                                           COMPRE  121
00474          END                                              COMPRE  122
```

```
                SUBROUTINE EFFECT(IPROG)                                        EFFECT    2
000003          COMMON/ALPHNU/O(63)/BLOCK/SPACE(14000),FAIL,FREE/DIGOP/IDIG,IOP,  BIGGER   11
               1IGOM,ISEMI,IOPI                                                 EFFECT    4
000003          INTEGER FAIL,FREE,PROG,RANGE,DOM,DOMAIN,DOMA,PAD1,HE,PAD2,SPACE,Q EFFECT    5
000003          CALL PLIST(IEQ,O(44))                                           EFFECT    6
000006          CALL PLIST(ILFT,O(41))                                          EFFECT    7
000011          CALL COPPY(IOP1,IOP,O)                                          EFFECT    8
000014          CALL INSERT(IOP1,O,Q(46))                                       EFFECT    9
000017          CALL INSERT(IOP1,O,Q(44))                                       COREFF    1
000022          CALL PLIST(IRB,O(50))                                           EFFECT   10
000025          HE=0                                                            EFFECT   11
000026          IDOMA=0                                                         EFFECT   12
000027          DO 1 K=1,99                                                     EFFECT   13
000031    1     CALL INSERT(HE,O,)(45))                                         EFFECT   14
000037          O(44)=0                                                         EFFECT   15
000040          PAGE=IPAGE                                                      EFFECT   16
000041          WRITE(6,2)                                                      EFFECT   17
000045    2     FORMAT(//,11(1H ),22HPROGRAM TO BE VERIFIED  ,                  EFFECT   18
               1  45(1H ),36HEFFECTIVE RANGE     EFFECTIVE DOMAIN ,/,          EFFECT   19
               211(1H ),22(1H-),45(1H ),15(1H-),5(1H ),16(1H-),//)             EFFECT   20
000046          GO TO 20                                                        EFFECT   21
                COPIN                                                           EFFECT   22
000047    10    CALL JOGTP(LINE,1,72)                                           COPI     86
000052          CALL DELETA(LINE)                                               EFFECT   24
                CEFFA                                                           EFFECT   25
000054    20    CALL ANY(PROG,ISEMI,K,0)                                        EFFECT   26
000057          IF(FAIL)21,21,199                                               EFFECT   27
000062    21    CALL COPPY(LINE,PROG,K)                                         EFFECT   28
000065          PAGE=SPACE(K+1)                                                 EFFECT   29
000067          IF(SPACE(ILFT)-O(43))22,10,22                                   EFFECT   30
000073    22    IF(SPACE(ILFT)-O(43))23,10,23                                   EFFECT   31
000076    23    CALL LEN(ILFT,7,K)                                              EFFECT   32
000101          IF(FAIL)24,24,10                                                EFFECT   33
000104    24    CALL ANY(K,IEQ,KK,0)                                            EFFECT   34
000107          IF(FAIL.EQ.1)GO TO 10                                           EFFECT   35
000112          CALL GAP(K,KK)                                                  EFFECT   36
000114          IF(FAIL.EQ.1)GO TO 10                                           EFFECT   37
000117          CALL COPPY(RANGE,K,KK)                                          EFFECT   38
000122          CALL ANY(RANGE,ILFT,KN,0)                                       COREFF1   1
000125          IF(FAIL)25,26,299                                               COREFF1   2
000130    25    CALL DELETA(RANGE,K1,0)                                         COREFF1   3
000133          CALL COPPY(DOM,K,0)                                             COREFF1   4
000136          GO TO 30                                                        COREFF1   5
000140    299   CALL COPPY(DOM,SPACE(KK+1),0)                                   COREFF1   6
                COPEA                                                           EFFECT   46
000144    30    CALL ANY(DOM,IEQ1,K,0)                                          EFFECT   47
000147          IF(FAIL)31,31,32                                                EFFECT   48
000152    31    CALL COPPY(DOMA,DOM,K)                                          EFFECT   49
000155          IPK12=0)1                                                       EFFECT   50
000157          CALL DELETE(DOM,IPK12,K)                                        EFFECT   51
000161          GO TO 34                                                        EFFECT   52
000163    32    IF(O )34,40,33                                                  EFFECT   53
000166    33    O(44)=0                                                         EFFECT   54
```

```
 000156          DOM=0                                      EFFECT
 000164   34     IF(DOMA)35,30,35                           EFFECT
 000167   35     CALL SPAN(DOMA,IDIG,K,KK,1)                EFFECT
 000173          IF(FAIL)30,0,36                            EFFECT
 000175   36     CALL INSERT(DOMA,0,Q(46))                  EFFECT
 000201          CALL STATUS(DOMAIN,DOMA,K,KK,0)            EFFECT
 000205          IF(FAIL)15,15,37                           EFFECT
 000210   15      IF(SPACE(KK+1).EQ.0)GO TO 30              EFFECT
 000212          KK=SPACE(KK+1)                             EFFECT
 000213          IF(SPACE(KK).EQ.Q(46))GO TO 30             EFFECT
 000215   37     IPK12=DOMAIN                               EFFECT
 000217          CALL CONCAT(DOMAIN,IPK12,DOMA)             EFFECT
 000221          CALL DELETB(DOMA)                          EFFECT
 000223          GO TO 30                                   EFFECT
              C EFFC                                        EFFECT
 000225   40     IF(SPACE(DOMAIN)-Q(46))42,41,42            EFFECT
 000230   41     CALL DELETA(DOMAIN,0)                      EFFECT
 000232   42     IPK12=                                     EFFECT
 000234          CALL CONCAT(RANGE,IPK12,IRB)               EFFECT
 000236          CALL INSERT(RANGE,0,Q(49))                 EFFECT
 000241          IPK12=DOMAIN                               EFFECT
 000243          CALL CONCAT(DOMAIN,IPK12,IRB)              EFFECT
 000245          CALL INSERT(DOMAIN,0,Q(49))                EFFECT
 000250          K=72-ISIZE(                                EFFECT
 000255          CALL LEN(   ,K,KK)                         EFFECT
 000257          CALL COPY(   1,HE,SPACE(KK+1))             EFFECT
 000263          K=1   ISIZE(    )+5                        EFFECT
 000270          CALL LEN(   ,K,KK)                         EFFECT
 000272          CALL COPY(PAD2,HE,SPACE(KK+1))             EFFECT
 000276          IPK12=                                     EFFECT
 000300          CALL CONCAT(LINE,IPK12,PAD1)               EFFECT
 000302          CALL DELETB(PAD1)                          EFFECT
 000304          CALL CONCAT(LINE,LINE,RANGE)               EFFECT
 000307          CALL DELETA(RANGE)                         EFFECT
 000311          CALL CONCAT(LINE,LINE,PAD2)                EFFECT
 000314          CALL DELETB(PAD2)                          EFFECT
 000316          CALL CONCAT(LINE,LINE,DOMAIN)              EFFECT
 000321          CALL DELETA(DOMAIN)                        EFFECT
 000323          CALL OUTPUT(LINE,1,130)                    EFFECT
 000326          CALL DELETA(LINE)                          EFFECT
 000330          GO TO 2                                    EFFECT
 000332  100     CALL DELETA(                               EFFECT
 000334          CALL DELETA(TOP1)                          EFFECT
 000336          CALL DELETA(LEFT)                          EFFECT
 000340          CALL DELETA(                               EFFECT
 000342          CALL DELETA(HE)                            EFFECT
 000344          RETURN                                     EFFECT
 000345          END                                        EFFECT
```

```
                INTEGER FUNCTION SIMPLI(IEXP)                    SIMPLI    2
000003          INTEGER PARSE,EVA                                SIMPLI    3
000003          IEXP=PARSE(IEXP)                                 SIMPLI    4
000006          SIMPLI=EVA(IEXP)                                 SIMPLI    5
000010          RETURN                                           SIMPLI    6
000010          END                                              SIMPLI    7
```

```
      SUBROUTINE PREC(IA,IB,K)                                          PREC      2
C  IF IA HAS A HIGHER OPERATOR PRECEDENCE THAN IB THEN K=1               PREC      3
C  IF IA HAS A LOWER OPERATOR PRECEDENCE THAN IB THEN K=-1               PREC      4
C  IF IA AND IB HAVE THE SAME OPERATOR PRECEDENCE THEN K=0               PREC      5
      INTEGER SPACE,ITABLE(57,2)                                        PREC      6
      COMMON/PTRN/ITABLE/CLOCK/SPACE(14000),IFQ,IFQQ                    BIGGER   12
      JA=-1                                                             PREC      8
      JB=-1                                                             PREC      9
      DO 1 J=1,57                                                       PREC     10
      IF(ITABLE(J,1).EQ.SPACE(IA))JA=ITABLE(J,2)                        PREC     11
      IF(ITABLE(J,1).EQ.SPACE(IB))JB=ITABLE(J,2)                        PREC     12
      IF(JA.NE.0.AND.JB.GT.0)GO TO 2                                    PREC     13
    1 CONTINUE                                                          PREC     14
    2 IF(JA-JB)3,4,5                                                    PREC     15
    3 K=-1                                                              PREC     16
      RETURN                                                            PREC     17
    4 K=0                                                               PREC     19
      RETURN                                                            PREC     19
    5 K=1                                                               PREC     20
      RETURN                                                            PREC     21
      END                                                               PREC     22
```

```
                      INTEGER FUNCTION PARSE(EXPRES)                                    PARSE     2
000003                INTEGER EXPRES,SPACE,FAIL,DIGITS,OP,XPX,XQX,PEND,A,B,C,DA,STRNG,D PARSE     3
                      1PRT,DP,DP,P                                                      PARSE     4
000013                COMMON/ALPHA/Q(45)/BLOCK/SPACE(14000),FAIL,IFOO                   BIGGER   13
000023                DIMENSION DIGITS(10),IOP,IQQ1,IQQ2,IQQ3                           PARSE     6
                      PEND=0                                                            PARSE     7
000033                A=0                                                               PARSE     8
000034                B=0                                                               PARSE     9
000035                C=0                                                               PARSE    10
000036                DA=0                                                              PARSE    11
000037                IPRT=0                                                            PARSE    12
000041                STRNG=0                                                           PARSE    13
000042                CALL COPY(OP,IOP,0)                                               PARSE    14
000014                CALL INSERT(OP,0,0(57))                                           PARSE    15
000017                CALL INSERT(OP,0,0(46))                                           PARSE    16
000022                CALL INSERT(OP,0,0(48))                                           PARSE    17
000025                CALL INSERT(OP,0,0(44))                                           PARSE    18
000031                CALL INSERT(OP,0,0(55))                                           PARSE    19
000033                CALL INSERT(OP,0,0(54))                                           PARSE    20
000036                CALL INSERT(OP,0,0(52))                                           PARSE    21
000041                CALL INSERT(OP,0,0(53))                                           PARSE    22
000044                CALL INSERT(OP,0,0(62))                                           PARSE    23
000047                CALL INSERT(OP,0,0(58))                                           PARSE    24
000052                CALL INSERT(OP,0,0(59))                                           PARSE    25
000055                CALL INSERT(OP,0,0(60))                                           PARSE    26
000060                CALL INSERT(OP,0,0(61))                                           PARSE    27
000063                CALL INSERT(OP,0,0(56))                                           PARSE    28
000066                CALL INSERT(OP,0,0(63))                                           PARSE    29
000071                CALL SLIST(IPAT,0(49))                                            PARSE    30
000074                CALL INSERT(IPAT,0,0(39))                                         PARSE    31
000077    2           CALL STRING(EXPRES,IPAT,K,KK,0)                                   PARSE    32
000103                IF(FAIL.EQ.1) GO TO 1                                             PARSE    33
000106                SPACE(K)=0(59)                                                    PARSE    34
000111                IPK12=      KK                                                    PARSE    35
000112                CALL DELETE(EXPRES,KK,IPK12)                                      PARSE    36
000116                GO TO 2                                                           PARSE    37
000119    1           CALL DELETE(IPAT)                                                 PARSE    38
000120                CALL SLIST(XPX,0(41))                                             PARSE    39
000123                CALL INSERT(XPX,0,0(42))                                          PARSE    40
000126                CALL INSERT(XPX,0,0(37))                                          PARSE    41
000131                CALL INSERT(XPX,0,0(38))                                          PARSE    42
000134                CALL INSERT(XPX,0,0(39))                                          PARSE    43
000137                CALL INSERT(XPX,0,0(40))                                          PARSE    44
000142                DO 3 I=1,3                                                        PARSE    45
000145                CALL INSERT(XPX,0,0(1))                                           PARSE    46
000150    3           CONTINUE                                                          PARSE    47
000153                CALL SLIST(XQX,0(44))                                             PARSE    48
000156                CALL INSERT(XQX,0,0(59))                                          PARSE    49
000161                CALL INSERT(XQX,0,0(59))                                          PARSE    50
000164                CALL INSERT(XQX,0,0(60))                                          PARSE    51
000167                CALL INSERT(XQX,0,0(61))                                          PARSE    52
000172                CALL SLIST(OP,0(47))                                              PARSE    53
000175                CALL INSERT(OP,0,0(51))                                           PARSE    54
```

```
000201          CALL INSERT(IP,,Q(14))                                  PARSE      55
000203          CALL INSERT(IP,,Q(47))                                  PARSE      56
000205          CALL NEXT(Q,,Q(57))                                     PARSE      57
000211          IPOINT=EXPRES                                           PARSE      58
000213   212    CALL ANY(IPOINT,XPX,K,Q)                                PARSE      59
000215          IF(FAIL.EQ.1)GO TO 219                                  PARSE      60
000221          IPOINT=SPACE(K+1)                                       PARSE      61
000223          CALL ANY(IPOINT,XOX,K,1)                                PARSE      62
000225          IF(FAIL.EQ.0.)GO TO 219                                 PARSE      63
000230          CALL STRING(IPOINT,NE,KK,K,1)                           PARSE      64
000232          IF(FAIL.EQ.1)GO TO 212                                  PARSE      65
000237   213    IPOINT=SPACE(K+1)                                       PARSE      66
000241          CALL SPAN(IPOINT,XPX,KK,K,1)                            PARSE      67
000246          IF(FAIL.EQ.1) GO TO 212                                 PARSE      68
000251          IPOINT=SPACE(K+1)                                       PARSE      69
000253          CALL ANY(IPOINT,XOX,KA,1)                               PARSE      70
000255          IF(FAIL.EQ.0)GO TO 215                                  PARSE      71
000257          CALL STRING(IPOINT,NE,KB,KA,1)                          PARSE      72
000263          IF(FAIL.EQ.1)GO TO 212                                  PARSE      73
000265   215    IPOINT=SPACE(KA+1)                                      PARSE      74
000271          CALL ANY(IPOINT,XPX,KA,1)                               PARSE      75
000273          IF(FAIL.EQ.1)GO TO 212                                  PARSE      76
000276          CALL COPY(X,EXPRES,K)                                   PARSE      77
000301          CALL COPY(1,XX,0)                                       PARSE      78
000304          CALL INSERT(0,0,X(35))                                  PARSE      79
000307          CALL CONCAT(EXPRES,A,B)                                 PARSE      80
000312          IPOINT=EXPRES                                           PARSE      81
000314          GO TO 22                                                PARSE      82
000316   213    CALL DELETE(XPX)                                        PARSE      83
000317          CALL DELETE(XOX)                                        PARSE      84
000321          IPOINT=EXPRES                                           PARSE      85
000323          CALL CONCAT(EXPRES,IPK12,DA)                            PARSE      86
                C 1000                                                  PARSE      87
000326          S(1)=)                                                  PARSE      88
000330          )                                                       PARSE      89
                C 1000                                                  PARSE      90
000335          IPOINT=EXPRES                                           PARSE      91
000341   221    DO 501 II=37,53                                         PARSE      92
000343          IF(SPACE(IPOINT+1).EQ.Q(II))GO TO 220                   PARSE      93
000346   501    CONTINUE                                                PARSE      94
000350          IPOINT=SPACE(IPOINT+1)                                  PARSE      95
000353          IF(ANY(IPOINT,,Q(41))GO TO 221                          PARSE      96
000360          SPACE(IPOINT)=Q(43)                                     PARSE      97
000365   220    IPOINT=SPACE(IPOINT+1)                                  PARSE      98
000367          IF(IPOINT)222,222,221                                   PARSE      99
000370   222    IF(SPACE(EXPRES).EQ.Q(33))SPACE(EXPRES)=Q(43)           PARSE     100
                C 1000                                                  PARSE     101
000375          CALL ASSIST(IPAT,,Q(37))                                PARSE     102
000377          CALL INSERT(IPAT,,Q(38))                                PARSE     103
000402          CALL INSERT(IPAT,,Q(39))                                PARSE     104
000405          CALL INSERT(IPAT,,Q(41))                                PARSE     105
000410          CALL INSERT(IPAT,,Q(40))                                PARSE     106
000413          CALL INSERT(IPAT,,Q(46))                                PARSE     107
000416          CALL INSERT(IPAT,,Q(48))                                PARSE     108
```

```
C00401          *CALL INSERT(IPAT,0,Q(47))                                      PARSE   109
C00404          CALL INSERT(IPAT,0,Q(48))                                       PARSE   110
C00407          CALL INSERT(IPAT,0,Q(52))                                       PARSE   111
C00412          CALL INSERT(IPAT,0,Q(53))                                       PARSE   112
C00415          CALL INSERT(IPAT,0,Q(54))                                       PARSE   113
C00420          CALL INSERT(IPAT,0,Q(55))                                       PARSE   114
C00423          CALL INSERT(IPAT,0,Q(58))                                       PARSE   115
C00426          CALL INSERT(IPAT,0,Q(59))                                       PARSE   116
C00431          CALL INSERT(IPAT,0,Q(60))                                       PARSE   117
C00434          CALL INSERT(IPAT,0,Q(61))                                       PARSE   118
C00437          CALL INSERT(IPAT,0,Q(62))                                       PARSE   119
C00442          IPOINT=EXPRES                                                   PARSE   120
C00446    223   CALL ANY(IPOINT,IPAT,K,0)                                       PARSE   121
C00447          IF(FAIL.EQ.1)GO TO 224                                          PARSE   122
C00452          IPOINT=SPACE(K+1)                                               PARSE   123
C00456          IF(SPACE(IPOINT).EQ.Q(38))SPACE(IPOINT)=Q(43)                   PARSE   124
C00461          GO TO 223                                                       PARSE   125
C00464    224   CALL DELETE(IPAT)                                               PARSE   126
                C EXP                                                           PARSE   127
C00473    250   CALL ANY(EXPRES,OP,K,1)                                         PARSE   128
C00477          IF(FAIL.EQ.1)GO TO 251                                          PARSE   129
C00502          CALL COPY(OPSTK,EXPRES,K)                                       PARSE   130
C00505          CALL DELETE(EXPRES,0)                                           PARSE   131
C00510          GO TO 250                                                       PARSE   132
C00513    251   CALL ANYF(EXPRES,OP,K,0)                                        PARSE   133
C00516          IF(FAIL.EQ.1)GO TO 252                                          PARSE   134
C00521          CALL DELETE(OPSTK,0)                                            PARSE   135
C00524          CALL COPY(OPSTK,EXPRES,K)                                       PARSE   136
C00527          CALL COPY(OPSTK,K,K)                                            PARSE   137
C00531          IPK12=EXPRES                                                    PARSE   138
C00533          CALL CONCAT(EXPRES,IPK12,K)                                     PARSE   139
C00536    252   IPK12=STACK                                                     PARSE   140
C00541          CALL CONCAT(STACK,IPK12,OP(NO))                                 PARSE   141
C00544          IPK12=STACK                                                     PARSE   142
C00547          CALL CONCAT(STACK,IPK12,OA)                                     PARSE   143
                C OPR                                                           PARSE   144
C00556    270   IF(SPACE(OPRTR).EQ.Q(44).OR.SPACE(OPRTR).EQ.Q(41))GO TO 290    PARSE   145
                C EXPRES                                                        PARSE   146
C00567    280   IF(K.EQ.0)GO TO 290                                             PARSE   147
C00567          KA=SPACE(K)                                                     PARSE   148
C00571          CALL POP(KA,OPSTK,K)                                            PARSE   149
C00576          IF(K)280,280,290                                                PARSE   150
                C OPR                                                           PARSE   151
C00605    290   IF(SPACE(OPRTR).EQ.Q(57))GO TO 500                              PARSE   152
C00611          CALL COPY(PA,OPSTR,0)                                           PARSE   153
C00615          CALL INSERT(4,0,KA)                                             PARSE   154
C00620          GO TO 500                                                       PARSE   155
                C SIX                                                           PARSE   156
C00623    300   KA=SPACE(5)                                                     PARSE   157
C00627          IF(SPACE(KA).EQ.Q(41)) GO TO 400                                PARSE   158
C00634          IPK12=STACK                                                     PARSE   159
C00637          CALL CONCAT(STACK,IPK12,KA)                                     PARSE   160
C00641          IPK12=STACK                                                     PARSE   161
C00644          CALL CONCAT(STACK,IPK12,OA)                                     PARSE   162
```

```
C00605          IF(SPACE(KA).EQ.0(48))GO TO 400                    PARSE   163
C00611         *CALL DELETA(KA)                                    PARSE   164
C00612          CALL DELETA(M,0)                                   PARSE   165
C00616          GO TO 240                                          PARSE   166
                                                                   PARSE   167
C     SEVEN                                                        PARSE   168
C00716    400   CALL DELETA(SPACE(KA))                            PARSE   169
C00721          CALL DELETA(M,0)                                   PARSE   170
C00725          GO TO 250                                          PARSE   171
C     TEN                                                          PARSE   172
C00750    500   PARSE=STATE                                        PARSE   173
C00757          CALL DELETA(OPRTR)                                 PARSE   174
C00763          CALL DELETA(OPEND)                                 PARSE   175
C00772          CALL DELETA(OPF)                                   PARSE   176
C00776          CALL DELETA(OPP)                                   PARSE   177
C01003          RETURN                                             PARSE   178
C01010          END                                                PARSE   179
```

```
              SUBROUTINE ACOMP(A11,B11)                                     ACOMP      2
000005        INTEGER SPACE,FAIL,DIGITS,A1,A11,B1,B11,PAT1,AAA,BBB,AAAA,BBBB,0   ACOMP      3
000005        COMMON/ZERO/0(63)/BLOCK/SPACE(14000),FAIL,IFQQ                BIGGER    14
000005        COMMON/DIGIT/DIGITS,IQQ1,IQQ2,IQQ3,IQQ4                       ACOMP      5
000005        PAT1 =0                                                       ACOMP      6
000005        AAA=0                                                         ACOMP      7
000006        BBB =0                                                        ACOMP      8
000010        CALL MLIST(ISTAR,0(39))                                       ACOMP      9
000012        CALL MLIST(AAA,0)                                             ACOMP     10
000014        CALL MLIST(BBB,0)                                             ACOMP     11
000016        CALL COPY(A1,A11,0)                                           ACOMP     12
000023        CALL COPY(B1,B11,0)                                           ACOMP     13
000027        CALL CONCAT(PAT1,DIGITS,ISTAR)                                ACOMP     14
000032        CALL CONCAT(A1,A1,ISTAR)                                      ACOMP     15
000035        CALL CONCAT(B1,B1,ISTAR)                                      ACOMP     16
000040        CALL ANY(A1,DIGITS,K,1)                                       ACOMP     17
000043        IF(FAIL.EQ.1)GO TO 100                                        ACOMP     18
000047        CALL SPAN(A1,PAT1,K,KK,1)                                     ACOMP     19
000052        CALL DELETE(A1,K,KK)                                          ACOMP     20
      C  SCAN                                                               ACOMP     21
000055  100   CALL ANY(B1,DIGITS,K,1)                                       ACOMP     22
000060        IF(FAIL.EQ.1)GO TO 200                                        ACOMP     23
000064        CALL SPAN(B1,PAT1,K,KK,1)                                     ACOMP     24
000067        CALL DELETE(B1,K,KK)                                          ACOMP     25
      C  [illegible]                                                        ACOMP     26
000072  200   CALL ANY(A1,ISTAR,K,0)                                        ACOMP     27
000075  202   IF(FAIL.EQ.1)GO TO 500                                        ACOMP     28
000101        CALL MAL2(A1,K)                                               ACOMP     29
000103        IF(FAIL.EQ.0)GO TO 201                                        ACOMP     30
000106        KK=SPACE(K+1)                                                 ACOMP     31
000110        CALL ANY(KK,ISTAR,K,0)                                        ACOMP     32
000113        GO TO 202                                                     ACOMP     33
000115  201   CALL DELETE(AAA)                                              ACOMP     34
000117        CALL COPY(AAA,A1,K)                                           ACOMP     35
000122        IPK12=A1                                                      ACOMP     36
000124        CALL DELETE(A1,IPK12,K)                                       ACOMP     37
000126        CALL ANY(B1,ISTAR,K,0)                                        ACOMP     38
000131  204   IF(FAIL.EQ.1)GO TO 600                                        ACOMP     39
000135        CALL MAL2(B1,K)                                               ACOMP     40
000137        IF(FAIL.EQ.0)GO TO 203                                        ACOMP     41
000142        KK=SPACE(K+1)                                                 ACOMP     42
000144        CALL ANY(KK,ISTAR,K,0)                                        ACOMP     43
000147        GO TO 204                                                     ACOMP     44
000151  203   CALL DELETE(BBB)                                              ACOMP     45
000153        CALL COPY(BBB,B1,K)                                           ACOMP     46
000156        IPK12=B1                                                      ACOMP     47
000160        CALL DELETE(B1,IPK12,K)                                       ACOMP     48
      C  [illegible]                                                        ACOMP     49
000162  300   IF(AAA.EQ.0)GO TO 400                                         ACOMP     50
000165        SPACE(AAAA)=SPACE(AAA)                                        ACOMP     51
000167        CALL DELETE(AAA,0)                                            ACOMP     52
000171        IF([illegible])GO TO 600                                      ACOMP     53
000174        SPACE([illegible])=SPACE(BBB)                                 ACOMP     54
```

```
000175           CALL DELETA(SPR,0)                              ACOMP   55
000200           CALL PROC(AAAA,BBBB,K)                          ACOMP   56
000207           IF(K)600,300,600                                ACOMP   57
      C   RCMP                                                   ACOMP   58
000207    400    IF(I.3.EQ.0)GO TO200                            ACOMP   59
000210    500    FAIL=0                                          ACOMP   60
000211           GO TO 700                                       ACOMP   61
000212    600    FAIL=1                                          ACOMP   62
000213    700    CALL DELETB(AAA)                                ACOMP   63
000215           CALL DELETB(BBB)                                ACOMP   64
000217           CALL DELETB(AAAA)                               ACOMP   65
000221           CALL DELETB(BBBB)                               ACOMP   66
000223           CALL DELETB(A1)                                 ACOMP   67
000225           CALL DELETB(B1)                                 ACOMP   68
000227           CALL DELETB(ISTAR)                              ACOMP   69
000231           CALL DELETB(PATL)                               ACOMP   70
000233           RETURN                                          ACOMP   71
000234           END                                             ACOMP   72
```

```
                 INTEGER FUNCTION STADD(STA3,STA2)                                   STADD     2
000006           INTEGER SPACE,FAIL,STA1,STA2,SUM,TERM1,TERM2,STA3,  SU1,SU2,FREE,Q  STADD     3
000015           COMMON/ALP2/AU7/Q(63)/BLOCK/SPACE(14000),FAIL,FREE                  BIGGER   15
000005           STA1 = STA3                                                         STADD     5
      C SET [illegible] TO NULL STRING, IE, SET TO ZERO.                             STADD     6
000005           SU1 = 0                                                             STADD     7
000006           TERM1 = 0                                                           STADD     8
000007           TERM2 = 0                                                           STADD     9
000010           SU1 = 0                                                             STADD    10
000011           SU2 = 0                                                             STADD    11
      C [illegible] TEMPORARY STRINGS                                                STADD    12
000012           CALL MLIST([illegible]Y,Q(43))                                      STADD    13
000015           CALL MLIST(IPLUS,Q(37))                                             STADD    14
000017           CALL MLIST([illegible]ANY,Q(37))                                    STADD    15
000023           CALL INSERT(MANY,0,Q(38))                                           STADD    16
000026           CALL INSERT([illegible]NY,1,Q(43))                                  STADD    17
      C TEST [illegible] BY CHECKING FOR NULL STRING                                 STADD    18
000031      2530 IF(STA1) 2550,2550,2540                                             STADD    19
      C [illegible] SUCCEEDED                                                        STADD    20
      C TEST [illegible] BY CHECKING FOR NULL STRING                                 STADD    21
000035      2540 IF (STA2) 2570,2560,2570                                            STADD    22
000037      2550 STADD=STA2                                                          STADD    23
[illegible]      STA2=0                                                              STADD    24
000041           GO TO 2775                                                          STADD    25
[illegible] 2560 STADD=STA1                                                          STADD    26
[illegible]      STA1=0                                                              STADD    27
[illegible]      GO TO 2775                                                          STADD    28
[illegible] 2570 IPK12=STA1                                                          STADD    29
[illegible]      [illegible](STA1,IPK12,[illegible]Y)                                STADD    30
[illegible]      IPK12=STA2                                                          STADD    31
[illegible]      CALL [illegible](STA2,IPK12,[illegible]Y)                           STADD    32
[illegible] 2590 IF(SPACE(STA1)-Q(33))2590,2610,2600                                 STADD    33
[illegible] 2600 IPK12=STA1                                                          STADD    34
[illegible]      CALL [illegible](STA1,IPLUS,IPK12)                                  STADD    35
[illegible] 2610 IF(SPACE(STA2)-Q(33))2620,2630,2620                                 STADD    36
[illegible] 2620 IPK12=STA2                                                          STADD    37
[illegible]      CALL [illegible](STA2,IPLUS,IPK12)                                  STADD    38
      C [illegible] ANY MATCH, THE BACK UP AND LOOK FOR BAL                          STADD    39
[illegible] 2630 ITEM = SPACE(STA1+1)                                                STADD    40
[illegible] 2632 CALL [illegible](ITEM,[illegible]NY,IDUM,0)                         STADD    41
[illegible]      IF(FAIL) [illegible],2632,2690                                      STADD    42
      C [illegible], BACK UP AND CHECK FOR BAL                                       STADD    43
[illegible] 2632 CALL BAL2(SPACE(STA1+1),IDUM)                                       STADD    44
000111           IF(FAIL) [illegible],2634                                           STADD    45
      C [illegible], LOOK FOR ANOTHER ANY MATCH                                      STADD    46
[illegible] 2634 ITEM= SPACE([illegible]+1)                                          STADD    47
[illegible]      [illegible]                                                         STADD    48
      C [illegible], MAKE ASSIGNMENTS OF SU1 AND TERM1                               STADD    49
[illegible] 2636 CALL COPY(SU1,STA1,STA1)                                            STADD    50
[illegible]      CALL COPY(TERM1,SPACE(STA1+1),IDUM)                                 STADD    51
000127           CALL [illegible]ST(TERM1)                                           STADD    52
      C [illegible] MAKE REPLACEMENT OF S19                                          STADD    53
[illegible]      ITEM=SPACE(IDUM)                                                    STADD    54
```

```
                      IPK12=STA1                                                STADD     55
                      CALL DELETE(STA1,IPK12,IDUM)                              STADD     56
                      CALL INSERT(STA1,0,ITEMP)                                 STADD     57
                C LOOK AHEAD FOR ANY MATCH, THE BACK UP AND LOOK FOR BAL        STADD     58
                2640  ITRY = SPACE(STA2+1)                                     STADD     59
                2641  CALL COMPARE(ITRY,TERM2,IDUM,0)                           STADD     60
                      IF(FAIL)2642,2642,2720                                    STADD     61
                C ANY MATCHES, BACK UP AND CHECK FOR BAL                        STADD     62
                2642  CALL BACK2(SPACE(STA2+1),IDUM)                            STADD     63
                      IF(FAIL)2643,2643,2644                                    STADD     64
                C BAL FAILED, LOOK FOR ANOTHER ANY MATCH                        STADD     65
                2643  ITRY=SPACE(IDUM+1)                                        STADD     66
                      GO TO 2641                                                STADD     67
                C COPY(IDUM), SAVE ASSIGNMENTS OF SU2 AND TERM2                 STADD     68
                2644  CALL COPY(SU2,STA2,STA2)                                  STADD     69
                      CALL COPY(TERM2,SPACE(STA2+1),IDUM)                       STADD     70
                      CALL LAST(TERM2)                                          STADD     71
                C DELETE, INSERT REPLACEMENT OF SU9                             STADD     72
                      ITRY=SPACE(IDUM)                                          STADD     73
                      IPAIR=STA2                                                STADD     74
                      CALL DELETE(STA2,IPK12,IDUM)                              STADD     75
                      CALL INSERT(STA2,0,ITEMP)                                 STADD     76
                2645  CALL PACK(TERM1,TERM2)                                    STADD     77
                      IF(IL) 2650,2650,2680                                     STADD     78
                2646  I=1/0                                                     STADD     79
                      CALL LOCATE(SU1,IPK12,SU1)                                STADD     80
                      IPK12=STA1                                                STADD     81
                      CALL LOCATE(SU1,IPK12,TERM1)                              STADD     82
                C LOOK AHEAD FOR ANY MATCH, THE BACK UP AND LOOK FOR BAL         STADD     83
                2671  ITRY = SPACE(STA1+1)                                      STADD     84
                2671  CALL COMPARE(ITRY,TERM1,IDUM,0)                           STADD     85
                      IF(FAIL) 2672,2672,2650                                   STADD     86
                2672  CALL BACK2(SPACE(STA1+1),IDUM)                            STADD     87
                      IF(FAIL) 2673,2676,2674                                   STADD     88
                C BAL FAILED, LOOK FOR ANOTHER ANY MATCH                        STADD     89
                2674  ITRY=SPACE(IDUM+1)                                        STADD     90
                      GO TO 2671                                                STADD     91
                C COPY(IDUM), SAVE ASSIGNMENTS OF SU1 AND TERM1                 STADD     92
                2676  CALL COPY(STA1,STA1,STA1)                                 STADD     93
                      CALL COPY(TERM1,SPACE(STA1+1),IDUM)                       STADD     94
                      CALL LAST(TERM1)                                          STADD     95
                C DELETE, INSERT REPLACEMENT OF SU9                             STADD     96
                      ITRY=SPACE(IDUM)                                          STADD     97
                      CALL DELETE(STA1,STA1,IDUM)                               STADD     98
                      CALL INSERT(STA1,0,ITEMP)                                 STADD     99
                      GO TO 2645                                                STADD    100
                2680  IPK12=SU2                                                 STADD    101
                      CALL LOCATE(SU2,IPK12,SU2)                                STADD    102
                      IPK12=STA2                                                STADD    103
                      CALL LOCATE(SU2,IPK12,TERM2)                              STADD    104
                      GO TO 2640                                                STADD    105
                2680  IF (IL+2) 2700,2700,2710                                  STADD    106
                2710  IPK12=SU1                                                 STADD    107
                      CALL LOCATE(SU1,IPK12,STA2)                               STADD    108
```

```
[illegible]                GO TO 2750                                   STADD   109
000322       2710   IPK12=S14                                    STADD   110
000324              CALL CONCAT(SUM,IPK12,SU2)                   STADD   111
[illegible]         IPK12=[illegible]                            STADD   112
[illegible]         CALL CONCAT(SUM,IPK12,TERM2)                 STADD   113
[illegible]         IPK12=[illegible]                            STADD   114
000334              CALL CONCAT(SUM,IPK12,STA2)                  STADD   115
[illegible]         GO TO 2750                                   STADD   116
[illegible]  2720   IF(I[illegible]1) 2730,2730,2740             STADD   117
[illegible]  2730   IPK12=[illegible]                            STADD   118
[illegible]         CALL CONCAT(SUM,IPK12,STA1)                  STADD   119
[illegible]         GO TO 2750                                   STADD   120
[illegible]  2740   IPK12=S14                                    STADD   121
[illegible]         CALL CONCAT(SUM,IPK12,SU1)                   STADD   122
[illegible]         IPK12=S14                                    STADD   123
[illegible]         CALL CONCAT(SUM,IPK12,TERM1)                 STADD   124
[illegible]         IPK12=[illegible]                            STADD   125
[illegible]         CALL CONCAT(SUM,IPK12,STA1)                  STADD   126
[illegible]  2750   IF ([illegible]) 2760,2760,2752              STADD   127
[illegible]  2752   IF([illegible](S1 ).EQ.[illegible](37))CALL DELETA(SUM,0)  STADD   128
       C [illegible] COPY [illegible] STADD                      STADD   129
[illegible]  2760   CALL EASE (SUM)                              STADD   130
[illegible]         STA[illegible]=S[illegible]                  STADD   131
[illegible]         S[illegible]=[illegible]                     STADD   132
       C [illegible] DELETE TEMPORARY STRINGS                    STADD   133
[illegible]  2776   CALL DELETA(DUMY,0)                          STADD   134
[illegible]         CALL DELETA(S14)                             STADD   135
000411              CALL DELETA(TERM1)                           STADD   136
[illegible]         CALL DELETA(TERM2)                           STADD   137
[illegible]         CALL DELETA(S11)                             STADD   138
[illegible]         CALL DELETA(S12)                             STADD   139
[illegible]         CALL DELETA(I[illegible]S,0)                 STADD   140
[illegible]         CALL DELETA(DUMY)                            STADD   141
[illegible]         RETURN                                       STADD   142
[illegible]         END                                          STADD   143
```

```
            INTEGER FUNCTION STMULT(AMU,BMU)                                  STMULT     2
000005      COMMON/ALPHNU/Q(63)/BLOCK/SPACE(14000),FAIL,IFOQ                  BIGGER    16
000005      INTEGER SPACE,FAIL,AMU,BMU,SIGNAL,TMULT,SIGN,SEG,TEMP,AMUL,BMUL,  STMULT     4
           * ST[illegible],PLUS,SS,TAIL,STA[illegible],Q                      STMULT     5
[illegible] ST[illegible]=0                                                   STMULT     5
[illegible] CALL [illegible]LIST([illegible]Y,Q(37))                          STMULT     7
000010      CALL INSERT([illegible]ANY,0,Q(38))                               STMULT     8
000013      [illegible]STAK=0                                                 STMULT     9
000014      IAMUST=0                                                          STMULT    10
[illegible] I[illegible]ST=0                                                  STMULT    11
[illegible] I[illegible]ST=0                                                  STMULT    12
[illegible] CALL [illegible]LIST(L[illegible]NUS,Q(38))                       STMULT    13
[illegible] [illegible] = Q(38)                                               STMULT    14
[illegible] CALL [illegible]LIST(PLUS,Q(37))                                  STMULT    15
[illegible] I[illegible]=Q(37)                                                STMULT    16
      C SET LOCALS TO ZERO                                                    STMULT    17
[illegible] 36[illegible] [illegible]=0                                       STMULT    18
[illegible] IF([illegible](AMU).EQ.Q(37))CALL DELETA(AMU,0)                   STMULT    19
[illegible] IF([illegible](BMU).EQ.Q(37))CALL DELETA(BMU,0)                   STMULT    20
[illegible] TMULT=0                                                           STMULT    21
[illegible] SI[illegible]=0                                                   STMULT    22
[illegible] [illegible]=0                                                     STMULT    23
[illegible] T[illegible]=0                                                    STMULT    24
[illegible] [illegible]IL=0                                                   STMULT    25
[illegible] [illegible]IL=0                                                   STMULT    26
[illegible] ST[illegible]T=0                                                  STMULT    27
[illegible] I[illegible]=SPACE(AMU+1)                                         STMULT    28
[illegible] 3610 CALL [illegible](I[illegible],[illegible]Y,I[illegible],0)   STMULT    29
[illegible] IF([illegible].[illegible].1)[illegible] TO 3620                  STMULT    30
[illegible] CALL [illegible]2([illegible],0)                                  STMULT    31
[illegible] IF([illegible].[illegible].0)[illegible] TO 3680                  STMULT    32
[illegible] I[illegible]=SPACE(I[illegible]+1)                                STMULT    33
[illegible] GO TO [illegible]                                                 STMULT    34
[illegible] 3620 I[illegible]=SPACE([illegible]+1)                            STMULT    35
[illegible] 3621 CALL [illegible](I[illegible],[illegible]Y,I[illegible],0)   STMULT    36
[illegible] IF([illegible].[illegible].1)GO TO 3630                           STMULT    37
[illegible] CALL [illegible]2(I[illegible],0)                                 STMULT    38
[illegible] IF([illegible].[illegible])GO TO 3690                             STMULT    39
[illegible] I[illegible]=SPACE(I[illegible]+1)                                STMULT    40
[illegible] [illegible]                                                       STMULT    41
[illegible] 3630 [illegible](AMU)                                             STMULT    42
[illegible] IF ([illegible]-I[illegible]ST) 3640,3632,3640                    STMULT    43
      C [illegible](0) [illegible]                                            STMULT    44
      C DELETE [illegible] ELEMENT OF AMU                                     STMULT    45
000121 3632 CALL [illegible](AMU,0)                                           STMULT    46
[illegible] GO TO [illegible]                                                 STMULT    47
[illegible] 3640 I[illegible] = [illegible](BMU)                              STMULT    48
[illegible] IF([illegible]-I[illegible]ST) 3650,3642,3650                     STMULT    49
      C DELETE THE FIRST ELEMENT OF BMU                                       STMULT    50
[illegible] 3642 CALL [illegible](BMU,0)                                      STMULT    51
000132      GO TO [illegible]                                                 STMULT    52
[illegible] 3650 STMULT = [illegible](AMU,BMU)                                STMULT    53
[illegible] [illegible]                                                       STMULT    54
```

```
000141   3660 ITEST = SPACE(BMU)                              STMULT   55
000142        IF(ITEST-MINUS) 3670,3662,3670                  STMULT   56
       C [illegible] (-) MATCHES                              STMULT   57
       C DELETE FIRST ELEMENT OF BMU                          STMULT   58
000145   3662 CALL DELETE(BMU,0)                              STMULT   59
000147        GO TO [illegible]                               STMULT   60
000151   3670 [illegible] = [illegible](SMU,[illegible])      STMULT   61
000154        CALL CONCAT(STMULT,LMINUS,IDUM)                 STMULT   62
000157        GO TO 3770                                      STMULT   63
000161   3680 ITEMP=SPACE([illegible]+1)                      STMULT   64
000163   3681 CALL [illegible](ITEMP,[illegible],IDUM,0)      STMULT   65
[illegible]   IF([illegible].0.1)GO TO 3720                   STMULT   66
000172        CALL [illegible](IDUM,0)                        STMULT   67
000174        IF([illegible].0.0)GO TO 3670                   STMULT   68
000177        ITEMP=SPACE(ITEMP+1)                            STMULT   69
000201        GO TO 3681                                      STMULT   70
000201   3690 [illegible]                                     STMULT   71
000202        [illegible]                                     STMULT   72
000203        [illegible]                                     STMULT   73
       C [illegible] FIRST ELEMENT OF [illegible] FOR = 1R-   STMULT   74
000204   3720 ITEST = SPACE([illegible])                      STMULT   75
000206        IF(ITEST-MINUS) 3730,3722,3730                  STMULT   76
       C CHECK SUCCESSFUL                                     STMULT   77
       C [illegible] DELETE FIRST CHAR. (-) OF BMU.           STMULT   78
000210   3722 [illegible] = [illegible]                       STMULT   79
000212        CALL DELETE ([illegible],0)                     STMULT   80
000214   3730 ITEST = SPACE (AMU)                             STMULT   81
000217        IF(ITEST-MINUS) 3740,3760,3740                  STMULT   82
000221   3740 ITEMP=AMU                                       STMULT   83
000222   3741 CALL [illegible](ITEMP,[illegible],IDUM,0)      STMULT   84
000225        IF([illegible].0.1)GO TO 3750                   STMULT   85
000231        CALL [illegible]([illegible],IDUM)              STMULT   86
000232        IF([illegible].0.0)GO TO 3742                   STMULT   87
000235        ITEMP=SPACE(ITEMP+1)                            STMULT   88
000237        GO TO 3741                                      STMULT   89
000237   3742 CALL [illegible]([illegible])                   STMULT   90
000243        CALL COPY([illegible],AMU,IDUM)                 STMULT   91
000246        CALL COPY([illegible],IDUM,0)                   STMULT   92
000251        CALL DELETE([illegible])                        STMULT   93
000254        [illegible]=[illegible]                         STMULT   94
000257   3750 [illegible]                                     STMULT   95
000261        CALL [illegible](STMULT,[illegible]K12,SIGN)    STMULT   96
000263        [illegible]([illegible],[illegible])            STMULT   97
000270        [illegible]                                     STMULT   98
000272        CALL CONCAT(STMULT,[illegible]K12,ITEMP)        STMULT   99
000274        CALL DELETE(ITEMP)                              STMULT  100
       C [illegible] AS POS(0)                                STMULT  101
000276   3760 IF([illegible])GO TO 3780                       STMULT  102
000300        IK=SPACE([illegible]+1)                         STMULT  103
000302   3762 CALL [illegible](IK,[illegible],IDUM,0)         STMULT  104
000305        IF([illegible].0.1)GO TO 3772                   STMULT  105
000311        CALL [illegible]([illegible](ITEM+1),IDUM)      STMULT  106
000314        IF([illegible].0.0)GO TO 3764                   STMULT  107
000317        IK=SPACE(ITEM+1)                                STMULT  108
```

```
000321        [illegible] GO TO 3762                                       STMULT
       C EVERYTHING MATCHED, MAKE ASSIGNMENTS OF SIGN, AMUL, SR            STMULT
000[illegible]  37[illegible] CALL COPY(SIGN,A[illegible],AMU)             STMULT
000324          CALL DELETE(AMUL)                                           STMULT
000326          CALL COPY([illegible]MUL,SPACE(AMU+1),IDUM)                 STMULT
000[illegible]  IPK1[illegible]=AMU                                         STMULT
000335          S[illegible]=SPACE(I[illegible])                            STMULT
000340          CALL DELETE(A[illegible],IPK12,IDUM)                        STMULT
000342          CALL INSERT([illegible],0,SR)                               STMULT
000346          GO TO 3790                                                  STMULT
       C LEN(1) SUCCEEDED ,ASSIGNMENT OF SIGN AND AMUL                      STMULT
000350  3772 CALL COPY(SI[illegible],A[illegible],AMU)                      STMULT
000353          KOPY = SPACE(AMU+1)                                         STMULT
000356          CALL COPY(AMUL,KOPY,0)                                      STMULT
       C [illegible] DELETE [illegible] THAT MATCHED                        STMULT
000361          CALL DELETE(AMU)                                            STMULT
000[illegible]4 [illegible]                                                 STMULT
       C IF SIGNAL IS [illegible]-NULL, IT WILL MATCH LEN(1)                STMULT
000365  37[illegible] IF(SIGNAL.[illegible].0)[illegible] TO 3860           STMULT
000367          IF(SPACE(I[illegible]T).EQ.Q(33))GO TO 3794                 STMULT
000372          CALL INSERT(TMULT,0,Q(38))                                  STMULT
000375          I[illegible]=SPACE(TMULT+1)                                 STMULT
000377          [illegible]                                                 STMULT
000[illegible]1 [illegible] 37[illegible]                                   STMULT
000[illegible]3  37[illegible] CALL DELETE(I[illegible]T,0)                 STMULT
000[illegible]  [illegible]                                                 STMULT
000[illegible]  [illegible]                                                 STMULT
000[illegible]  37[illegible] CALL [illegible]Y([illegible],[illegible]Y,IDUM,0) STMULT
000[illegible]3 IF([illegible].EQ.1)GO TO 3860                              STMULT
000[illegible]7 CALL [illegible]                                            STMULT
000[illegible]1 IF(FAIL.[illegible])GO TO 3830                              STMULT
000[illegible]  [illegible]+1)                                              STMULT
000[illegible]  [illegible] 37[illegible]                                   STMULT
000[illegible]  38[illegible] IF([illegible].[illegible](38))GO TO 3840     STMULT
000[illegible]  [illegible]=Q(33)                                           STMULT
000[illegible]  [illegible] 3750                                            STMULT
000[illegible]3  38[illegible] S[illegible]=Q(37)                           STMULT
000[illegible]5  3850 [illegible]=SPACE(I[illegible]+1)                     STMULT
000[illegible]7 [illegible]                                                 STMULT
000441          GO TO 37[illegible]                                         STMULT
000441  38[illegible] STMULT=TMULT                                          STMULT
000442          T[illegible]=3                                              STMULT
000443          [illegible] 37[illegible]                                   STMULT
000446  38[illegible] IFEST = I[illegible]                                  STMULT
000447          JFEST = I[illegible]                                        STMULT
000452          IF(IFEST-JFEST)3880,3910,3910                               STMULT
       C S[illegible]                                                       STMULT
000[illegible]  38[illegible] [illegible]                                   STMULT
000457          A[illegible]                                                STMULT
000457          [illegible]=IFE[illegible]                                  STMULT
000[illegible]  3910 IF([illegible]) 3912,3911,3912                         STMULT
       C LEN(1) FAILS                                                       STMULT
000461  3911 GO TO 3[illegible]                                             STMULT
       C LEN(1) SUCCEEDED, LOOK FOR + OR -, THEN GO BACK AND CHECK FOR BAL  STMULT
```

```
000662   3912  ITEMP=SPACE(ISUB+1)                                             STMULT
000664   3913  CALL ANY(ITEMP,SANY,ITRY,0)                                     STMULT
000667         IF(FAIL) 3914,3914,3920                                         STMULT
         C BACKING UP AND CHECKING FOR BAL                                     STMULT
000673   3914  CALL BAL2(14J,ITRY)                                             STMULT
000675         IF (FAIL) 3917,3917,3915                                        STMULT
         C BAL FAILED, TRY ANOTHER + AND -                                     STMULT
000701   3915  ITEMP = SPACE(ITRY+1)                                           STMULT
000703         GO TO 3913                                                      STMULT
         C POS(J) [illegible] BAL ANY  HAVE MATCHED, MAKE ASSIGNMENTS OF BMUL AND STRD   STMULT
000704   3917  CALL [illegible](BMUL)                                          STMULT
000706         CALL COPY(BMUL,[illegible],ITRY)                                STMULT
         C SPACE(CAT) = + [illegible] -                                        STMULT
000712         [illegible] = SPACE(ITRY)                                       STMULT
000714         [illegible]                                                     STMULT
000716         CALL [illegible],IPK12,ITRY)                                    STMULT
000722         IF([illegible]) 3918,3918,3919                                  STMULT
         C IF [illegible] REPLACEMENT OF STRD = -.                             STMULT
000724   3918  CALL INSERT([illegible],0,0(33))                                STMULT
000727         GO TO [illegible]                                               STMULT
         C [illegible] STRD = +                                                STMULT
000731   3919  CALL INSERT([illegible],0,0(37))                                STMULT
         C [illegible] HERE ******************                                 STMULT
000735   3920  CALL COPY(ITRY,[illegible],0)                                   STMULT
000737         CALL INSERT([illegible],0,ITRY)                                 STMULT
000741         CALL INSERT([illegible],0,[illegible])                          STMULT
000743         CALL [illegible],0,[illegible] )                                STMULT
000745         CALL [illegible],0,3)                                           STMULT
000747         [illegible]                                                     STMULT
000750         [illegible]                                                     STMULT
000752   3931  [illegible]                                                     STMULT
000754   [illegible] CALL [illegible],IBACK,1)                                 STMULT
000757         IF([illegible]) [illegible],3931,3960                           STMULT
         C [illegible] FOR NEXT ANY MATCH, THEN COME BACK FOR BAL              STMULT
000763   [illegible]                                                           STMULT
000765         [illegible] = [illegible]+1)                                    STMULT
000767   3932  CALL [illegible],ITRY,0)                                        STMULT
000771         IF (FAIL) [illegible],3960                                      STMULT
         C [illegible] BAL                                                     STMULT
000775   3933  CALL [illegible]2([illegible],ITRY)                             STMULT
000777         IF(FAIL) [illegible],3934                                       STMULT
         C [illegible] -                                                       STMULT
001013   3934  [illegible](ITRY+1)                                             STMULT
001015         GO TO 3932                                                      STMULT
         C [illegible], MAKE ASSIGNMENTS OF BAU AND STRD                       STMULT
001016   3 [illegible] CALL COPY([illegible],ITRY)                             STMULT
001021         CALL [illegible]                                                STMULT
         C [illegible] MATCHED OR + OR -                                       STMULT
001022         [illegible] = SPACE(ITRY)                                       STMULT
001024         [illegible]                                                     STMULT
001026         CALL [illegible],IPK12,ITRY)                                    STMULT
001031         IF([illegible]) 3935,3937,3936                                  STMULT
         C IT WAS A PLUS                                                       STMULT
001034   3935  CALL INSERT(330,0,0(37))                                        STMULT
```

```
000637            GO TO 3940                                   STMULT   217
          C IT WAS A  I IS                                     STMULT   218
000641    3937  CALL INSERT(SHU,0,J(38))                       STMULT   219
000644    3940 ITEST = SPACE(BMU)                              STMULT   220
000647          IF (ITEST-IPLUS) 3950,3942,3950                STMULT   221
          C POS(1) WAS MATCHED, DELETE FIRST CHAR. OF BMUL     STMULT   222
000651    3942 CALL DELETA(BMU,0)                              STMULT   223
          C  ******WE ARE RECURSING HERE *******************  STMULT   224
000653    3950  CALL COPPY(ITRY,AMU,0)                         STMULT   225
000660          CALL INSERT(IAMUST,0,ITRY)                     STMULT   226
000665          CALL INSERT(IBMUST,0,BMU)                      STMULT   227
000670          CALL INSERT(ITMUST,0,TMULT)                    STMULT   228
000673          CALL INSERT(ISTAK,0,1)                         STMULT   229
000676          RETURN                                         STMULT   230
000701          GO TO 3999                                     STMULT   231
000701    3951   TMULT=STADD(TMULT,ISTMUL)                     STMULT   232
000704          GO TO 3930                                     STMULT   233
000706    3960 ITEST = SPACE(AMU)                              STMULT   234
000707          IF(ITEST-IPLUS) 3970,3962,3970                 STMULT   235
          C POS(1) WAS MATCHED, MAKE DELETION                  STMULT   236
000712    3962 CALL DELETA(AMU,0)                              STMULT   237
          C  ******WE ARE RECURSING HERE *******************  STMULT   238
000714    3970  CALL COPPY(KP,AMU,0)                           STMULT   239
000721          CALL INSERT(IAMUST,0,KP)                       STMULT   240
000724          CALL COPPY(KP,BMU,0)                           STMULT   241
000730          CALL INSERT(IBMUST,0,KP)                       STMULT   242
000733          CALL INSERT(ITMUST,0,TMULT)                    STMULT   243
000736          CALL INSERT(ISTAK,0,2)                         STMULT   244
000741          GO TO 3999                                     STMULT   245
000743    3971   STMULT=STADD(TMULT,ISTMUL)                    STMULT   246
          C  DELETION OF TEMPORARIES                           STMULT   247
000746    3975  [illegible] 3981,3980                          STMULT   248
000751    3980  CALL [illegible]                               STMULT   249
000753          [illegible](IAMUST)                            STMULT   250
000755          CALL DELETA(IAMUST,0)                          STMULT   251
000757          CALL DELETA(BMU)                               STMULT   252
000762          ISPACE(IBMUST)                                 STMULT   253
000765          CALL DELETA(IBMUST,0)                          STMULT   254
000767          ITEST=SPACE(ITMUST)                            STMULT   255
000771          CALL DELETA(ITMUST,0)                          STMULT   256
000773          ITEST=SPACE(ISTAK)                             STMULT   257
000776          CALL DELETA(ISTAK,0)                          STMULT   258
000777          [illegible]                                    STMULT   259
001000          IF(ITEST-1) 3982,3951,3982                     STMULT   260
001004    3982  IF(ITEST-2) 3921,3971,3921                     STMULT   261
001007    3981  CALL DELETA(PLUS,0)                            STMULT   262
001011          CALL DELETA(MINUS,0)                           STMULT   263
001013          CALL DELETA(AMU)                               STMULT   264
001016          CALL DELETA(BMU)                               STMULT   265
001021          CALL DELETA(ANY)                               STMULT   266
001023          CALL DELETA(SIGN)                              STMULT   267
001025          RETURN                                         STMULT   268
001027          END                                            STMULT   269
```

```
                      INTEGER FUNCTION COLLEC(LEX)                                  COLLEC     2
000003                COMMON/ALPHNO/Q(63)/BLOCK/SPACE(14000),FAIL,IFQQ              BIGGER    17
000033                COMMON/DIGIT/PATTRN,IQQ1,IQQ2,IQQ3,IQQ4                       COLLEC     4
000037                INTEGER SPACE,PLOC,BGN,END,FAIL,PLUS,COSIG1,ENDA,COSIG2       COLLEC     5
000053                INTEGER ENDA,PATTRN,VALUE,STAR,COEF1,ENDC,COEF2,TEMP1,TEMP2   COLLEC     6
000073                INTEGER DIT,DECODE,ENCODE,COLO ,Q                             COLLEC     7
                C  SET UP INITIAL STRINGS                                           COLLEC     8
000103                COLO = 0                                                      COLLEC     9
000104                COEF1=0                                                       COLLEC    10
000105                COEF2=0                                                       COLLEC    11
000106                COSIG1=0                                                      COLLEC    12
000107                COSIG2=0                                                      COLLEC    13
000110                DIT=0                                                         COLLEC    14
000111                LEX1=0                                                        COLLEC    15
000112                LEX2=0                                                        COLLEC    16
000113                CALL PLIST(STAR,Q(39))                                        COLLEC    17
000115                CALL PLIST(PLUS,Q(37))                                        COLLEC    18
000120                CALL PLIST(IPMO,Q(37))                                        COLLEC    19
000123                CALL INSERT(IPMO,1,Q(38))                                     COLLEC    20
000127                CALL INSERT(IPMO,2,Q(57))                                     COLLEC    21
000131                CALL PLIST(PLOC,Q(57))                                        COLLEC    22
000136                CALL PLIST(MINUS,Q(38))                                       COLLEC    23
                C  LEX = LEX **                                                     COLLEC    24
000137                IPK12=LEX                                                     COLLEC    25
000141                CALL CONCAT(LEX,IPK12,PLOC)                                   COLLEC    26
                C  LEX POS(0) *-*                                                   COLLEC    27
000143                CALL STRING(LEX,MINUS,BGN,END,1)                              COLLEC    28
000147                IF(FAIL) 1,2,1                                                COLLEC    29
                C  LEX = *** LEX                                                    COLLEC    30
000152          1     IPK12=LEX                                                     COLLEC    31
000155                CALL CONCAT(LEX,PLUS,IPK12)                                   COLLEC    32
                C  C LEFT = **                                                      COLLEC    33
000160            2 CALL COPY(0                                                   COLLEC    34
                                                                                    COLLEC    35
                C  LEX POS(0) ('0'(1) . COSIGN1 *AL . LEX1 ANY(*+-**) . DIT = DIT   COLLEC    36
                C                                                                   COLLEC    37
000165            40 IPK1=LEX                                                     COLLEC    38
000167            55 IF(IPK1) 3,20,3                                              COLLEC    39
000201             3 CALL ATT(SPACE(IPK1+1),IPMO,IND,0)                           COLLEC    40
000215               IF(FAIL) 21,4,20                                             COLLEC    41
000220             4 CALL VAL(SPACE(LEX+1),IND)                                   COLLEC    42
000243               IF(FAIL) 5,50,5                                              COLLEC    43
000246             5 IPK1=LEX                                                     COLLEC    44
000253               GO TO 55                                                     COLLEC    45
000260          50   CALL COPY(C STG1,LEX,LEX)                                    COLLEC    46
000303               CALL ADD(LEX1,SPACE(LEX+1),IND)                              COLLEC    47
000317               CALL COPY(LEX1,LEX,LEX)                                      COLLEC    48
000332               IPK12=LEX                                                    COLLEC    49
000335               CALL CONCAT(LEX,IPK12,IND)                                   COLLEC    50
000344               IPK12=LEX                                                    COLLEC    51
000350               CALL CONCAT(LEX,DIT,IPK12)                                   COLLEC    52
                C  LEX1 POS(0) SPAN('0123456789') . COEF1 = **                      COLLEC    53
000362               CALL SPAN(C,LEX1,PATTRN,BGN,ENDA,1)                          COLLEC    54
```

```
000126          IF(FAIL) 7,,,7                                          COLLEC     55
000131        * CALL COPY(COEF1,LEX1,ENDA)                              COLLEC     56
000134          LEX1=SPACE(ENDA+1)                                      COLLEC     57
000138          GO TO 8                                                 COLLEC     58
              C  LEX1 = *** LEX1                                        COLLEC     59
000143    7     IPK12=LEX1                                              COLLEC     60
000142          CALL CONCAT(LEX1,STAR,IPK12)                            COLLEC     61
              C  COEF1 = 1                                              COLLEC     62
000144          CALL ASSIGN(COEF1,0(28))                                COLLEC     63
              C  LEX POS(0) LEX(1) . COSIGN2 BAL . LEX2 ANY('+-*') . COLD = COLD  COLLEC  64
000147        * INDX=LEX                                                COLLEC     65
000151          IF(INDX) 10,19,10                                       COLLEC     66
000152    10    CALL ANY(SPACE(INDX+1),IPMD,IND,0)                      COLLEC     67
000157          IF(FAIL) 14,12,19                                       COLLEC     68
000161    12    CALL SIZE2(SPACE(LEX+1),IND)                            COLLEC     69
000166          IF(FAIL) 54,55,54                                       COLLEC     70
000173        * INDX=IND                                                COLLEC     71
000171          GO TO 11                                                COLLEC     72
000171    55    CALL COPY(COSIGN2,LEX,LEX)                              COLLEC     73
000176          CALL SIZE2(LEX2,SPACE(LEX+1),IND)                       COLLEC     74
000200          CALL COPY(COLD,IND,IND)                                 COLLEC     75
000203          IPK12=LEX                                               COLLEC     76
000205          CALL CONCAT(LEX,IPK12,IND)                              COLLEC     77
000207          IPK12=LEX                                               COLLEC     78
000211          CALL CONCAT(LEX,COLD,IPK12)                             COLLEC     79
              C  LEX2 POS(0) SPAN('0123456789A') . COEF2 = **           COLLEC     80
000215          CALL SPAN(LEX2,PATTRN,BIN,ENDA,1)                       COLLEC     81
000217          IF(FAIL) 16,13,14                                       COLLEC     82
000221    13    CALL COPY(COEF2,BIN,ENDA)                               COLLEC     83
000225          IPK12=LEX2                                              COLLEC     84
000227          CALL CONCAT(LEX2,IPK12,ENDA)                            COLLEC     85
000231          GO TO 15                                                COLLEC     86
              C  LEX2 = *** LEX2                                        COLLEC     87
000233    14    IPK12=LEX2                                              COLLEC     88
000235          CALL CONCAT(LEX2,STAR,IPK12)                            COLLEC     89
              C  COEF2 = 1                                              COLLEC     90
000237          CALL ASSIGN(COEF2,0(28))                                COLLEC     91
              C  LEX1 POS(0) LEX2 RPOS(0)                               COLLEC     92
000242    15    CALL IDENT(LEX1,LEX2)                                   COLLEC     93
000246          IF(FAIL) 16,16,19                                       COLLEC     94
              C  COEF1 = (-1)**(COEF1) * (COSIGN2 COEF2)                COLLEC     95
              C  CONVERT TO INTEGERS COEF1 AND COEF2 TO NUMBERS         COLLEC     96
000247    16    IPK1=DATA(COEF1)                                        COLLEC     97
000251          IPK2=DATA(COEF2)                                        COLLEC     98
              C  COMPLEMENT SIGNS IF SIGNS ARE MINUS                    COLLEC     99
000256          CALL IDENT(COSIGN1,MINUS)                               COLLEC    100
000260          IF(FAIL) 21,17,21                                       COLLEC    101
000261    17    IPK1=-IPK1                                              COLLEC    102
000263    21    CALL IDENT(COSIGN2,MINUS)                               COLLEC    103
000264          IF(FAIL) 23,22,23                                       COLLEC    104
000267    22    IPK2=-IPK2                                              COLLEC    105
              C  ADD THE TWO TEMPORARYS                                 COLLEC    106
000273    23    IPK1=IPK1+IPK2                                          COLLEC    107
              C  CONVERT TOTAL RESULT BACK TO STRING FORM               COLLEC    108
```

```
000272          COEF1=ENCODE(TEMP1)                                     COLLEC  109
        C  IDENT(COEF1,0)                                               COLLEC  110
000273          IF(TEMP1) 24,40,24                                      COLLEC  111
        C  COEF1 = POS(0) *-* . COSIGN1 = **                            COLLEC  112
000276       24 CALL SIGN(COEF1,MINUS,BGN,END,1)                        COLLEC  113
000332          IF(FAIL) 30,25,30                                       COLLEC  114
000335       25 CALL COPY(COSIG1,COEF1,COLF1)                           COLLEC  115
000340          COEF1=SPACE(COEF1+1)                                    COLLEC  116
000342          GO TO 9                                                 COLLEC  117
        C  COSIGN1 = ***                                                COLLEC  118
000343       30 CALL ALIST(COSIG1,Q(37))                                COLLEC  119
000346          GO TO 9                                                 COLLEC  120
        C  COLLECT = COLLECT COSIGN1 COEF1 LEX1                         COLLEC  121
000371       10 IPK12=COLLEC                                            COLLEC  122
000422          CALL CONCAT(COLLEC,IPK12,COSIG1)                        COLLEC  123
000424          CALL CONCAT(COLLEC,COLLEC,COEF1)                        COLLEC  124
000427          CALL CONCAT(COLLEC,COLLEC,LEX1)                         COLLEC  125
        C  COSIG2 = COSIGN2                                             COLLEC  126
000442          CALL COPY(COSIG1,COSIG2,0)                              COLLEC  127
        C  COEF1 = COEF2                                                COLLEC  128
000445          CALL COPY(COEF1,COEF2,0)                                COLLEC  129
        C  LEX1 = LEX2                                                  COLLEC  130
000460          CALL COPY(LEX1,LEX2,0)                                  COLLEC  131
000463          GO TO 9                                                 COLLEC  132
        C  COLLECT = COLLECT COSIGN1 COEF1 LEX1                         COLLEC  133
000465       10 IPK12=COLLEC                                            COLLEC  134
000467          CALL CONCAT(COLLEC,IPK12,COSIG1)                        COLLEC  135
000351          IPK12=      COLLEC                                      COLLEC  136
000353          CALL CONCAT(COLLEC,IPK12,COEF1)                         COLLEC  137
000355          IPK12=COLLEC                                            COLLEC  138
000357          CALL CONCAT(COLLEC,IPK12,LEX1)                          COLLEC  139
        C  COLLEC POS(0) *** = **                                       COLLEC  140
000361       27 IF(SAME(COLLEC).EQ.Q(37))CALL DELETA(COLLEC,0)          COLLEC  141
000370       32 CALL DELETS(STAR)                                       COLLEC  142
000372          CALL DELETS(PLUS)                                       COLLEC  143
000374          CALL DELETS(MINUS)                                      COLLEC  144
000376          CALL DELETS(IPK)                                        COLLEC  145
000400          CALL DELETS(PLUS)                                       COLLEC  146
000402          CALL DELETS(LEX)                                        COLLEC  147
000404          CALL DELETS(COEF1)                                      COLLEC  148
000406          CALL DELETS(COEF2)                                      COLLEC  149
000410          CALL DELETS(COSIG1)                                     COLLEC  150
000412          CALL DELETS(COSIG2)                                     COLLEC  151
000414          CALL DELETS(I)                                          COLLEC  152
000416          CALL DELETS(LEX1)                                       COLLEC  153
000420          CALL DELETS(LEX2)                                       COLLEC  154
000422          CALL DELETS(COLLEC)                                     COLLEC  155
000424          RETURN                                                  COLLEC  156
000425          END                                                     COLLEC  157
```

```
                INTEGER FUNCTION REDUND(ST)                                        REDUND    2
000003          COMMON/ [illegible] /BLOCK/SPACE(14000),FAIL,IFREE                 BIGGER   18
000003          INTEGER SPACE,FAIL,ZSTAR,OSTAR,ST,PLUS,PPNTR,A,BGN,END,PM,ZERO     REDUND    4
000003          INTEGER BGNA,BGNB,TEST,PREV,Q                                      REDUND    5
          C  SET UP INITIAL STRINGS                                                REDUND    6
000003          REDUND=0                                                           REDUND    7
000003          A=0                                                                REDUND    8
000004          ITEMP=0                                                            REDUND    9
          C  ZSTA  = #0*#                                                          REDUND   10
000006          CALL MLIST(ZSTAR,Q(27))                                            REDUND   11
000010          CALL INSERT(ZSTAR,1,Q(39))                                         REDUND   12
          C  OSTAR = #1*#                                                          REDUND   13
000013          CALL MLIST(OSTAR,Q(28))                                            REDUND   14
000016          CALL INSERT(OSTAR,1,Q(39))                                         REDUND   15
          C  PM = #+-#                                                             REDUND   16
000021          CALL MLIST(PM,Q(37))                                               REDUND   17
000024          CALL INSERT(PM,1,Q(38))                                            REDUND   18
          C  ZERO = #0#                                                            REDUND   19
000027          CALL MLIST(ZERO,Q(27))                                             REDUND   20
000032          CALL MLIST(PLUS,Q(38))                                             REDUND   21
000035          CALL MLIST(PLUS,Q(37))                                             REDUND   22
          C  REDUND = ST #+#                                                       REDUND   23
000040          CALL CONCAT(REDUND,ST,PLUS)                                        REDUND   24
          C  REDUND POS(0) #-#                                                     REDUND   25
000044          CALL STRIPS(REDUND,MINUS,BGN,END,1)                                REDUND   26
000050          IF(FAIL) 1,2,1                                                     REDUND   27
          C  REDUND = ### REDUND                                                   REDUND   28
000053   1      IPK12=[illegible]                                                  REDUND   29
000055          CALL CONCAT(REDUND,PLUS,IPK12)                                     REDUND   30
          C  REDUND ANY(#+-#) ### ANY(#+-#) . A = A                                 REDUND   31
000057        2 [illegible]                                                        REDUND   32
000061       50 IF(INDX) 3,4,3                                                     REDUND   33
000064        3 CALL ANY(INDX,PM,BGN,0)                                            REDUND   34
000067          IF(FAIL) 5,4,5                                                     REDUND   35
000072        4 CALL STRIPS(SPACE(BGN+1),ZERO,BGNA,END,1)                          REDUND   36
000077          IF(FAIL) 60,51,60                                                  REDUND   37
000102       51 CALL ANY(SPACE(END+1),PM,BGNB,1)                                   REDUND   38
000106          IF(FAIL) 60,52,60                                                  REDUND   39
000111   52     [illegible] SPACE([illegible])                                     REDUND   40
000113          [illegible]                                                        REDUND   41
000114       60 CALL DUP(ITEMP,REDUND,BGN)                                         REDUND   42
000117          TEXT [illegible]                                                   REDUND   43
000121          CALL [illegible](REDUND,IPK12,END)                                 REDUND   44
000123          IPK12=[illegible]                                                  REDUND   45
000125          CALL CONCAT(REDUND,ITEMP,IPK12)                                    REDUND   46
000127          CALL DELETS(ITEMP)                                                 REDUND   47
000131          [illegible]                                                        REDUND   48
          C  REDUND ANY(#+-#) #0*# BAL ANY(#+-#) . DUNO = DUNO                     REDUND   49
000133        6 IND=REDUND                                                         REDUND   50
000135       30 IF(IND) 7,10,7                                                     REDUND   51
000135        7 CALL ANY(IND,PM,IND1,0)                                            REDUND   52
000141          IF(FAIL) 10,8,10                                                   REDUND   53
000144        8 CALL STRIPS(SPACE(IND1+1),ZSTAR,IND2,IND3,1)                       REDUND   54
```

```
000151          IF(FAIL) 40,31,40                                          REDUND
000154       31 INDX=IND3                                                  REDUND
000156       32 CALL ANY(SPACE(INDX+1),PM,IND4,0)                          REDUND
000162          IF(FAIL) 40,33,40                                          REDUND
000165       33 CALL [illegible](SPACE(IND3+1),IND4)                       REDUND
000170          IF(FAIL) 45,34,45                                          REDUND
000173       45 INDX=IND4                                                  REDUND
000175          GO TO 32                                                   REDUND
000176   40     IND=SPACE(IND1+1)                                          REDUND
000177          GO TO 30                                                   REDUND
000200       34 CALL DUP(ITEMP,REDUND,IND1)                                REDUND
000203          IDUN=SPACE(IND)                                            REDUND
000205          IPK12=REDUND                                               REDUND
000207          CALL DELETE(REDUND,IPK12,IND4)                             REDUND
000211          CALL INSERT(REDUND,0,IDUN)                                 REDUND
000214          IPK12=[illegible]                                          REDUND
000215          CALL CONCAT(REDUND,ITEMP,IPK12)                            REDUND
000220          CALL DELETB(ITEMP)                                         REDUND
000222          GO TO 6                                                    REDUND
        C                                                                  REDUND
        C  REDUND ANY(*+-*) . A *1** = A                                   REDUND
        C                                                                  REDUND
000223       10 IND=REDUND                                                 REDUND
000225       91 IF(IND) 30,13,30                                           REDUND
000227       30 CALL ANY(IND,PM,IND1,0)                                    REDUND
000232          IF(FAIL) 13,11,13                                          REDUND
000235       11 CALL STERM(SPACE(IND1+1),0STAR,IND2,IND3,1)                REDUND
000242          IF(FAIL) 21,12,21                                          REDUND
000245       21 IND=SPACE(IND1+1)                                          REDUND
000247          GO TO 91                                                   REDUND
000250       12 CALL COPY(ITEMP,REDUND,IND1)                               REDUND
000253          IPK12=[illegible]                                          REDUND
000255          CALL DELETE(REDUND,IPK12,IND3)                             REDUND
000257          IPK12=REDUND                                               REDUND
000261          CALL CONCAT(REDUND,ITEMP,IPK12)                            REDUND
000263          CALL DELETB(ITEMP)                                         REDUND
000265          GO TO 10                                                   REDUND
        C  REDUND [illegible]S(0) *** = **                                 REDUND
000267   13     IF([illegible](REDUND).EQ.Q(37))CALL DELETA(REDUND,0)      REDUND
        C  REDUND ANY(*+-*) RPOS(0) = **                                   REDUND
000275          IF([illegible].-1.0)GOTO 19                                REDUND
000277          CALL LAST(REDUND)                                          REDUND
        C  REDUND [illegible](1)                                           REDUND
000300       18 CALL [illegible](REDUND,1,END)                             REDUND
000303          IF(FAIL) 19,20,19                                          REDUND
        C  REDUND = 0                                                      REDUND
000305   19     CALL DELETB(REDUND)                                        REDUND
000310          CALL INSERT(REDUND,Q(27))                                  REDUND
        C  CLEAR ALL STRINGS BUT REDUND TO MAKE MORE ROOM IN SPACE         REDUND
000313   20     CALL DELETB(2STAR)                                         REDUND
000315          CALL DELETB(S1)                                            REDUND
000317          CALL DELETB(0STAR)                                         REDUND
000321          CALL DELETB(PLUS)                                          REDUND
000323          CALL DELETB(MINUS)                                         REDUND
```

289

```
000325          CALL DELETA(PM)                          REDUND   109
000329          CALL DELETS(ZERO)                        REDUND   110
000331          RETURN                                   REDUND   111
000333          END                                      REDUND   112
```

```
              INTEGER FUNCTION SHORT(EXP)                                          SHORT     2
000003        COMMON /ALPHBT/Q(63)/BLOCK/SPACE(14000),FAIL,IFQQ                    BIGGER   19
000003        INTEGER RAND2,TEST,COUNT,ARROW,PFACT,BGN,OP2,SHORT6,END,ENCODE,Q     SHORT     4
000003        INTEGER EXP,PLOC,INDA,ENDA,OP1,FAIL,SPACE,ENDC,LOC,RAND1,SHOP        SHORT     5
        C   SET UP INITIAL STRINGS                                                 SHORT     6
000003        SHORT=0                                                              SHORT     7
000003        SHORT6=0                                                             SHORT     8
000004        COUNT=0                                                              SHORT     9
000005        OP1=0                                                                SHORT    10
000006        OP2=0                                                                SHORT    11
000007        RAND1=0                                                              SHORT    12
000010        RAND2=0                                                              SHORT    13
000011        ITEM=0                                                               SHORT    14
000012        SHOP=0                                                               SHORT    15
        C   PLOC=*+-*                                                              SHORT    16
000014        CALL MLIST(PLOC,Q(37))                                               SHORT    17
000016        CALL INSERT(PLOC,1,Q(38))                                            SHORT    18
        C   TEST = ***                                                             SHORT    19
000021        CALL MLIST(TEST,Q(39))                                               SHORT    20
        C   ARROW = ***                                                            SHORT    21
000024        CALL MLIST(ARROW,Q(56))                                              SHORT    22
        C   IPMS = *+-*                                                            SHORT    23
000027        CALL MLIST(IPMS,Q(37))                                               SHORT    24
000032        CALL INSERT(IPMS,1,Q(38))                                            SHORT    25
000035        CALL INSERT(IPMS,2,Q(39))                                            SHORT    26
        C                                                                          SHORT    27
        C   EXP [illegible] (ANY(*+-*) . OP1 BAL . RAND1 // BAL . RAND1) ANY(*+-**) .  SHORT    28
        C   . SHOP = S[illegible]                                                  SHORT    29
        C                                                                          SHORT    30
000040        CALL ANY(EXP,PLOC,INDA,1)                                            SHORT    31
000044        IF(FAIL) 2,1,2                                                       SHORT    32
000047      1 IF(INDA) 52,2,52                                                     SHORT    33
000050     52 CALL ANY(SPACE(INDA+1),IPMS,IND2,0)                                  SHORT    34
000056        IF(FAIL) 2,50,2                                                      SHORT    35
000057     50 CALL [illegible]2(SPACE(EXP+1),IND2)                                 SHORT    36
000062        IF(FAIL) 51,40,51                                                    SHORT    37
000065     51 INDA=IND2                                                            SHORT    38
000067        GO TO 1                                                              SHORT    39
000067      2 INDA=EXP                                                             SHORT    40
000070     53 IF(INDA) 53,4,53                                                     SHORT    41
000071     53 CALL ANY(IPMS,IPMS,IND2,0)                                           SHORT    42
000074        IF(FAIL) 4,54,4                                                      SHORT    43
000077     54 CALL [illegible](EXP,IND2)                                           SHORT    44
000102        IF(FAIL) 55,41,55                                                    SHORT    45
000104     55 [illegible]=SPACE(IND2+1)                                            SHORT    46
000105        GO TO 53                                                             SHORT    47
        C   SHORT = EXP                                                            SHORT    48
000107      4 CALL COPY(SHORT,EXP,0)                                               SHORT    49
000112        GO TO 26                                                             SHORT    50
000114     40 CALL COPY(OP1,EXP,EXP)                                               SHORT    51
000117        CALL DUP(RAND1,SPACE(EXP+1),IND2)                                    SHORT    52
000123        GO TO 42                                                             SHORT    53
000125     41 CALL DUP(RAND1,EXP,IND2)                                             SHORT    54
```

```
000130      42 CALL COPY(SHOP,IND2,IND2)                                    SHORT    55
000135         IPK12=EXP                                                    SHORT    56
000135         CALL DELETE(EXP,IPK12,IND2)                                  SHORT    57
000137         IPK12=EXP                                                    SHORT    58
000141         CALL CONCAT(EXP,SHOP,IPK12)                                  SHORT    59
            C  EXP POS(0) LEN(1) . OP2 BAL . RAND2 (ANY(*+-*/) // RPOS(0)) . SHOP =  SHORT    60
000143      6  IPKX=EXP                                                     SHORT    61
000145         IF(EXP) 7,30,7                                               SHORT    62
000146      7  IF(INDX) 8,60,8                                              SHORT    63
000147      8  CALL ANY(SPACE(INDX+1),IPMS,IND,0)                           SHORT    64
000153         IF(FAIL) 60,9,60                                             SHORT    65
000155      9  CALL CAL2(SPACE(EXP+1),IND)                                  SHORT    66
000161         IF(FAIL) 61,10,61                                            SHORT    67
000164      61 INDX=IND                                                     SHORT    68
000166         GO TO 7                                                      SHORT    69
000167      10 CALL COPY(OP2,EXP,EXP)                                       SHORT    70
000171         CALL COPY(RAND2,SPACE(EXP+1),IND)                            SHORT    71
000175         CALL COPY(ITEMP,1),IND)                                      SHORT    72
000200         EXP=SPACE(IND+1)                                             SHORT    73
000203         IPK12=EXP                                                    SHORT    74
000204         CALL CONCAT(EXP,ITEMP,IPK12)                                 SHORT    75
000215         GO TO 100                                                    SHORT    76
000210      60 CALL CAL2(SPACE(EXP+1),0)                                    SHORT    77
000213         IF(FAIL) 30,141,30                                           SHORT    78
000214      141 CALL COPY(OP2,EXP,EXP)                                      SHORT    79
000221         CALL COPY(RAND2,SPACE(EXP+1),0)                              SHORT    80
000225         CALL DELETE(EXP)                                             SHORT    81
            C  OP2 ***                                                      SHORT    82
000227      100 CALL STRING(OP2,TEST,BGN,END,0)                             SHORT    83
000233         IF(FAIL) 15,12,15                                            SHORT    84
            C  RAND1 POS(0) RAND2 RPOS(0)                                   SHORT    85
000235      12 CALL INT(RAND1,RAND2)                                        SHORT    86
000240         IF(FAIL) 15,13,15                                            SHORT    87
            C  COUNT = COUNT + 1                                            SHORT    88
000243      13 COUNT=COUNT+1                                                SHORT    89
000245         GO TO 5                                                      SHORT    90
            C  ST(COUNT,*)                                                  SHORT    91
000246      15 IF(COUNT) 24,24,16                                           SHORT    92
            C                                                               SHORT    93
            C  SHORT = SHORT OP1 RAND1 *** (COUNT + 1)                      SHORT    94
            C                                                               SHORT    95
            C  SHORT = SHORT OP1                                            SHORT    96
000247      16    IPK12=SHORT                                               SHORT    97
000251         CALL CONCAT(SHORT,IPK12,OP1)                                 SHORT    98
            C  SHORT = SHORT RAND1                                          SHORT    99
000253         IPK12=SHORT                                                  SHORT   100
000255         CALL CONCAT(SHORT,IPK12,RAND1)                               SHORT   101
            C  SHORT = SHORT ***                                            SHORT   102
000257         IPK12=SHORT                                                  SHORT   103
000261         CALL CONCAT(SHORT,IPK12,ARROW)                               SHORT   104
            C  SHORT = SHORT (COUNT + 1)                                    SHORT   105
000263         COUNT=COUNT+1                                                SHORT   106
000265         SHORT=SHORT//=(COUNT)                                        SHORT   107
000266         IPK12=SHORT                                                  SHORT   108
```

```
                      CALL CONCAT(SHORT,IPK12,PFACT)                    SHORT     109
                C     IDENT(SHORT6,1)                                   SHORT     110
                      IF(SHORT6-1) 17,26,17                             SHORT     111
                C     CPLIT = 0                                         SHORT     112
                17    CPLIT=1                                           SHORT     113
                C     OP1 = 0.2                                         SHORT     114
                      IF (CALL COPY(OP1,OP2,0)                          SHORT     115
                C     RAND1 = RAND2                                     SHORT     116
                      CALL COPY(RAND1,RAND2,0)                          SHORT     117
                      GO TO 6                                           SHORT     118
                C     SHORT = SUBST OP1 RAND1                           SHORT     119
                24      IPK12=SHORT                                     SHORT     120
                      CALL CONCAT(SHORT,IPK12,OP1)                      SHORT     121
                      IPK12=SHORT                                       SHORT     122
                      CALL CONCAT(SHORT,IPK12,RAND1)                    SHORT     123
                C     IDENT(SHORT6,1)                                   SHORT     124
                      IF(SHORT6-1) 18,26,18                             SHORT     125
                25    CALL DELETE(OP1)                                  SHORT     126
                      CALL DELETE(OP2)                                  SHORT     127
                      CALL DELETE(RAND1)                                SHORT     128
                      CALL DELETE(RAND2)                                SHORT     129
                      CALL DELETE(PLUS)                                 SHORT     130
                      CALL DELETE(TEST)                                 SHORT     131
                      CALL DELETE(ARROW)                                SHORT     132
                      CALL DELETE(IPKS)                                 SHORT     133
                      CALL DELETE(SHORT)                                SHORT     134
                      RETURN                                            SHORT     135
                C     SHORT = 1                                         SHORT     136
                      DO SHORT =1                                       SHORT     137
                      GO TO 15                                          SHORT     138
                      END                                               SHORT     139
```

```
             FUNCTION MERGE(MERP1,MERP2)                                      MERGE    2
000005       COMMON/ALPH/IZ2(63)/BLOCK/IFREE(14000),FAIL,FREE                  BIGGER  20
000005       COMMON/DIGIP/DIGITS,IQQ1,IQQ2,IQQ3,IQQ4                           MERGE    4
000005       INTEGER FAIL,FREE,CONS1,CONS2,TERM1,TERM2,DIGITS,ENCODE,DECODE,Q  MERGE    5
000005       CALL COPY(MERP1,MERP1,0)                                          MERGE    6
000010       CALL COPY(MERP2,MERP2,0)                                          MERGE    7
000014       KSTR=0                                                            MERGE    8
      C INITIALIZE THE LOCAL VARIABLES                                         MERGE    9
000015           CONS1 = 0                                                     MERGE   10
000015           CONS2 = 0                                                     MERGE   11
000016          TERM1 = 0                                                      MERGE   12
000017          TERM2 = 0                                                      MERGE   13
000020          MERG=0                                                         MERGE   14
000022          CALL SLIST(ISTR,Q(39))                                         MERGE   15
000024          CALL SLIST(JSTR,Q(43))                                         MERGE   16
000027          CALL CONCAT(KSTR,ISTR,JSTR)                                    MERGE   17
      C MERG      MER1 = *** MER1 ***                                          MERGE   18
000032          CALL INSERT (MER1,0,Q(39))                                     MERGE   19
000035          CALL CONCAT(MER1,MER1,JSTR)                                    MERGE   20
      C           MER2 = *** MER2 ***                                          MERGE   21
000040          CALL INSERT(MER2,0,Q(39))                                      MERGE   22
000043          CALL CONCAT(MER2,MER2,JSTR)                                    MERGE   23
      C           MER1 POS(0) LEN(1) SPAN(*0123456739*) . CONS1 = **   /F(PME)  MERGE   24
000046       IF(MER1)33,1,33                                                   MERGE   25
000051  33      IT = IFREE(MER1+1)                                             MERGE   26
000053       CALL SPAN(IT,DIGITS,KA,KB,1)                                      MERGE   27
000057       IF(FAIL)2,2,1                                                     MERGE   28
000063  2     CALL REPLACE(MER1,1,0)                                           MERGE   29
000066        MER1=IFREE(MER1)                                                 MERGE   30
000067       IFREE(KB+1)=0                                                     MERGE   31
000071        CONS1=DECODE(KA)                                                 MERGE   32
000073       CALL DELETE(KA)                                                   MERGE   33
      C           MER2 POS(0) LEN(1) SPAN(*0123456798*) . CONS2 = **   /F(OME)  MERGE   34
000076       IF(MER2)34,4,34                                                   MERGE   35
000100  34      IT = IFREE(MER2+1)                                             MERGE   36
000102       CALL SPAN(IT,DIGITS,KA,KB,1)                                      MERGE   37
000106       IF(FAIL)4,5,4                                                     MERGE   38
000112  5    CALL DELETE(MER2,0)                                               MERGE   39
000114       MER2=IFREE(KB+1)                                                  MERGE   40
000116       IFREE(KB+1)=0                                                     MERGE   41
000117       CONS2=DECODE(KA)                                                  MERGE   42
000122       CALL DELETE(KA)                                                   MERGE   43
      C           MERG = CONS1 * CONS2    /(NME)                               MERGE   44
000125  500   MERG=ENCODE(CONS1*CONS2)                                         MERGE   45
000130       GO TO 7                                                           MERGE   46
      C OME       MERG = CONS1    /(NME)                                       MERGE   47
000133  4    MERG=ENCODE(CONS1)                                                MERGE   48
000135       GO TO 7                                                           MERGE   49
      C PME       MER2 POS(0) LEN(1) SPAN(*0123456789*) . MERG =               MERGE   50
000137  1       IF(MER2)7,7,35                                                 MERGE   51
000141  35    CALL SPAN(IFREE(MER2+1),DIGITS,KA,KB,1)                          MERGE   52
000145       IF(FAIL)3,3,7                                                     MERGE   53
000152  3    CALL COPY(MERG,KA,KB)                                             MERGE   54
```

214

```
000155          IPK12=MER2                                                        MERGE    55
000157          CALL DELETE(MER2,IPK12,KB)                                        MERGE    56
         C XME  MER1 POS(0) LEN(1) BAL . TER41 ANY(**$*) . CONS1 = CONS1   /F(XME    MERGE    57
000161   7          IF(MER1)11,11,36                                              MERGE    58
000165   36         II = IFREE(MER1+1)                                            MERGE    59
000167   11         CALL ANY(II,KSTR,KK,0)                                        MERGE    60
000172              IF(FAIL)15,15,11                                              MERGE    61
000176   15           CALL BAL2(IFREE(MER1+1),KK)                                 MERGE    62
000201              IF(FAIL)17,17,16                                              MERGE    63
000205   16     KK=IFREE(KK+1)                                                    MERGE    64
000207              GO TO 14                                                      MERGE    65
000212   17         CALL COPY(TERM1,IFREE(MER1+1),KK)                             MERGE    66
000214   12         CALL DELETA(MER1,0)                                           MERGE    67
000216              IF(MER1-KK)12,13,12                                           MERGE    69
         C YME  MER2 POS(0) LEN(1) BAL . TER42 ANY(**$*) . CONS1 = CONS1   /F(YME    MERGE    69
000221   13         IF(MER2)19,19,37                                              MERGE    70
000223   37         II = IFREE(MER2+1)                                            MERGE    71
000225   24         CALL ANY(II,KSTR,KK,0)                                        MERGE    72
000230              IF(FAIL)25,25,19                                              MERGE    73
000234   25         CALL BAL2(IFREE(MER2+1),KK)                                   MERGE    74
000237              IF(FAIL)27,27,26                                              MERGE    75
000243   26     KK=IFREE(KK+1)                                                    MERGE    76
000245              GO TO 24                                                      MERGE    77
000246   27         CALL COPY(TERM2,IFREE(MER2+1),KK)                             MERGE    78
000252   22         CALL DELETA(MER2,0)                                           MERGE    79
000254              IF(MER2-KK)22,28,22                                           MERGE    80
         C TME          IDENT(TERM1,TERM2)    /F(TME)                             MERGE    81
000257   28         CALL IDENT(TERM1,TERM2)                                       MERGE    82
000261              IF(FAIL)22,32,29                                              MERGE    83
         C              MERG = MERG *** TER41                                     MERGE    84
000265   32     IPK12=MERG                                                        MERGE    85
000267          CALL CONCAT(MERG,IPK12,ISTR)                                      MERGE    86
000271          IPK12=MERG                                                        MERGE    87
000273          CALL CONCAT(MERG,IPK12,TERM1)                                     MERGE    88
000275          CALL DELETB(TERM1)                                                MERGE    89
         C XME  MER1 POS(0) LEN(1) BAL . TER41 ANY(**$*) . CONS1 = CONS1 /F(XME)S(RM   MERGE    90
000277              IF(MER1)11,11,46                                              MERGE    91
000303   46         II = IFREE(MER1+1)                                            MERGE    92
000305   44     CALL ANY(II,KSTR,KK,0)                                            MERGE    93
000310              IF(FAIL)45,45,11                                              MERGE    94
000314   45           CALL BAL2(IFREE(MER1+1),KK)                                 MERGE    95
000317              IF(FAIL)47,47,43                                              MERGE    96
000323   43     KK=IFREE(KK+1)                                                    MERGE    97
000325              GO TO 44                                                      MERGE    98
000326   47         CALL COPY(TERM1,IFREE(MER1+1),KK)                             MERGE    99
000332   42         CALL DELETA(MER1,0)                                           MERGE   100
000334              IF(MER1-KK)42,28,42                                           MERGE   101
         C TME          MERG = MERG *** TERM2   /(OME)                            MERGE   102
000340   29     IPK12=MERG                                                        MERGE   103
000342          CALL CONCAT(MERG,IPK12,ISTR)                                      MERGE   104
000344          IPK12=MERG                                                        MERGE   105
000346          CALL CONCAT(MERG,IPK12,TERM2)                                     MERGE   106
000350          CALL DELETB(TERM2)                                                MERGE   107
000352              GO TO 18                                                      MERGE   108
```

```
            C XME        TERM2 LEN(1)     /S(ZME)                       MERGE
000354      11           IF(TERM2)39,40,39                              MERGE
            C              MERG = MERG MER2     /(ZME)                  MERGE
000355      40      IPK12=MERG                                          MERGE
000357              CALL CONCAT(MERG,IPK12,MER2)                        MERGE
000361              CALL DELETS(MER2)                                   MERGE
000363                GO TO 41                                          MERGE
            C WME         MERG = MERG *** TERM2 MER2     /(ZME)         MERGE
000366      39      IPK12=MERG                                          MERGE
000367              CALL CONCAT (MERG,IPK12,ISTR)                       MERGE
000371              IPK12=MERG                                          MERGE
000373              CALL CONCAT(MERG,IPK12,TERM2)                       MERGE
000375              CALL DELETS(TERM2)                                  MERGE
000377              IPK12=MERG                                          MERGE
000401              CALL CONCAT(MERG,IPK12,MER2)                        MERGE
000403              CALL DELETS(MER2)                                   MERGE
000405                GO TO 41                                          MERGE
            C YME        TERM1 LEN(1)     /S(VME)                       MERGE
000407      10           IF(TERM1)42,40,48                              MERGE
            C              MERG = MERG MER1     /(ZME)                  MERGE
000411      40       IPK12=MERG                                         MERGE
000412                CALL CONCAT(MERG,IPK12,MER1 )                     MERGE
000414              CALL DELETS(MER1)                                   MERGE
000416                GO TO 41                                          MERGE
            C VME          MERG = MERG *** TERM1 MER1                   MERGE
000420      42        IPK12=MERG                                        MERGE
000422              CALL CONCAT(MERG,IPK12,ISTR)                        MERGE
000424              IPK12=MERG                                          MERGE
000426              CALL CONCAT(MERG,IPK12,TERM1)                       MERGE
000430              CALL DELETS(TERM1)                                  MERGE
000432                 IPK12=MERG                                       MERGE
000434              CALL CONCAT(MERG,IPK12,MER2     )                   MERGE
000436              CALL DELETS(MER2)                                   MERGE
            C ZME         MERG POS(0) *** = //                          MERGE
000437      41           IF(IPK12(MERG)-0(33))51,52,51                  MERGE
000444      52           CALL DELETA(MERG,0)                            MERGE
            C              MERG NEAR(***) . MERGE     /(RETURN)         MERGE
000446      51           CALL SNY(MERG,JSTR,10,0)                       MERGE
000451                   CALL COPY(MERGE,MERG,1M)                       MERGE
            C  DELETE THE STRINGS THAT ARE NO LONGER NEEDED              MERGE
000455                   CALL DELETS(MERG)                              MERGE
000457                   CALL DELETS(ISTR)                              MERGE
000461                   CALL DELETS(JSTR)                              MERGE
000463                   CALL DELETS(KSTR)                              MERGE
000465              CALL DELETS(MER1)                                   MERGE
000466              CALL DELETS(MER2)                                   MERGE
000470                 RETURN                                           MERGE
000472                 END                                              MERGE
```

```
                 INTEGER FUNCTION EVA(POLISH)                                         EVA      2
000003           COMMON/ALPHA/Q(63)/BLOCK/IFREE(14000),FAIL,FREE                      BIGGER  21
000003           COMMON/DIGIT/DIGITS,I001,I002,I003,I004                              EVA      4
000003           INTEGER POLISH,FAIL,FREE,DOWAR,SUBJEC,C,STMULT,EXPO,TERM,TTERM,      EVA      5
                C,SUBJCT,MULT,ST1,ST2,DIGITS,ENCODE,DECODE,STADD,COLLEC,REDIND,       EVA      6
                CSHORT,D                                                              EVA      7
000003           DATA JTRUE/2HTR/,JFALSE/2HFA/                                        COR6    20
000003           CALL NLIST(DOWAR,Q(57))                                              EVA      8
000006           CALL NLIST(MIN1,Q(28))                                               EVA      9
000011           CALL INSERT(MIN1,0,Q(38))                                            EVA     10
000014           CALL NLIST(IADD,Q(55))                                               EVA     11
000017           CALL NLIST(ICOM,Q(46))                                               EVA     12
000022           CALL NLIST(IRP,Q(42))                                                EVA     13
000025           CALL NLIST(IPAT15,Q(37))                                             EVA     14
000030           CALL INSERT(IPAT15,0,Q(38))                                          EVA     15
000033           CALL INSERT(IPAT15,0,Q(39))                                          EVA     16
000036           CALL INSERT(IPAT15,0,Q(40))                                          EVA     17
000041           CALL NLIST(ILP,Q(41))                                                EVA     18
000044           CALL NLIST(IPAT10,Q(44))                                             EVA     19
000047           CALL INSERT(IPAT10,0,Q(52))                                          EVA     20
000052           CALL INSERT(IPAT10,0,Q(58))                                          EVA     21
000055           CALL INSERT(IPAT10,0,Q(59))                                          EVA     22
000060           CALL INSERT(IPAT10,0,Q(60))                                          EVA     23
000063           CALL INSERT(IPAT10,0,Q(61))                                          EVA     24
000066           CALL NLIST(IANY,Q(37))                                               COR9     1
000071           CALL INSERT(IANY,0,Q(38))                                            COR9     2
000074           CALL NLIST(ITRUE,JTRUE)                                              COR6    21
000077           CALL NLIST(IFALSE ,JFALSE)                                           COR6    22
000102           IALPH=0                                                              EVA     38
000103           DO 1 I=1,26                                                          EVA     39
000104           CALL INSERT(IALPH,0,Q(I))                                            EVA     40
000107    1      CONTINUE                                                             EVA     41
000112           CALL NLIST(IPAT11,Q(41))                                             EVA     42.
000115           CALL INSERT(IPAT11,0,Q(42))                                          EVA     43
000120           CALL INSERT(IPAT11,0,Q(44))                                          EVA     44
000123           CALL CONCAT(IPAT11,IPAT11,DIGITS)                                    EVA     45
000126           CALL CONCAT(IPAT11,IPAT11,IALPH)                                     EVA     46
000131           SUBJEC=0                                                             EVA     47
000132   100     C=0                                                                  EVA     48
000133   200     CALL ANY(POLISH,DOWAR,KA,0)                                          EVA     49
000137           IF(FAIL)201,201,2700                                                 EVA     50
000142   201     CALL DELETB(SUBJEC)                                                  EVA     51
000144           CALL COPY(SUBJEC,POLISH,KA)                                          EVA     52
000150           IPK12=POLISH                                                         EVA     53
000152           CALL DLETB(POLISH,IPK12,KA)                                          EVA     54
000154           DO 202 II = 37,63                                                    EVA     56
000157           IF(ITOP(SUBJEC).EQ.Q(II))GO TO 300                                   EVA     57
000162   202     CONTINUE                                                             EVA     58
000165           CALL INSERT(C,0,SUBJEC)                                              COR2     1
000166           SUBJEC=0                                                             EVA     60
000167           GO TO 200                                                            EVA     61
000171   300     IF(ITPLEF(SUBJEC)-Q(43))400,301,400                                  EVA     62
000174   301     CALL COPY(IMIN,MIN1,0)                                               EVA     63
```

```
000177          KA=STORLT(ISIGN,IFREE(C))                                   EVA   64
000202          IFREE(C) = KA                                               EVA   65
000205          GO TO 200                                                   EVA   66
000206    400   IF(IFREE(STORE)-N(56))700,402,700                           EVA   67
000211    402   EXPO=IFREE(C)                                               EVA   68
000213          CALL DELETA(C,0)                                            EVA   69
000215          TERM=IFREE(C)                                               EVA   70
000217    500   KA=SIGCODE(EXPO)                                            COR2   2
000222          CALL DELETA(EXPO)                                           COR2   3
000223          IF(KA)501,502,503                                           COR2   4
000226    501   WRITE(6,504)KA                                              COR2   5
000234    504   FORMAT(1H0,*NEGATIVE EXPONENT*  ,I15, 20H ASSUMED TO BE 0****  )  COR2   6
000254    502   CALL SETST(ITERM,0(28))                                     COR2   7
000257          GO TO 506                                                   COR2   8
000261    503   CALL COPY(TTERM,TERM,0)                                     COR2   9
000264    505   IF(KA.EQ.1)GO TO 506                                        COR2  10
000267          KA=KA-1                                                     COR2  11
000270          TTERM=STORLT(TERM,TTERM)                                    COR2  12
000253          GO TO 505                                                   COR2  13
000254    506   IFREE(C)=TTERM                                              COR2  14
000256          TTERM=0                                                     COR2  15
000257          CALL DELETA(TERM)                                           COR2  16
000261          GO TO 200                                                   COR2  17
000262    700   IF(IFREE(STORE)-N(39))800,701,800                           EVA   82
000265    701   AMULT=IFREE(C)                                              EVA   83
000267          CALL DELETA(C,0)                                            EVA   84
000271          BMULT=IFREE(C)                                              EVA   85
000273          IFREE(C)=STORLT(AMULT,BMULT)                                EVA   86
000277          GO TO 200                                                   EVA   87
000301    800   IF(STORE(STORE-C)-N(37))1000,801,1000                       EVA   88
000304    801   ST1=IFREE(C)                                                EVA   89
000307          CALL DELETA(C,0)                                            EVA   90
000311          ST2=IFREE(C)                                                EVA   91
000312          IFREE(C)=STORD(ST1,ST2)                                     EVA   92
000316          CALL DELETA(ST1)                                            EVA   93
000320          CALL DELETA(ST2)                                            EVA   94
000322          GO TO 200                                                   EVA   95
000324    1000  IF(STORE(STORE-C)-N(38))1100,1001,1100                      EVA   96
000327    1001  CALL COPY(TERM,MULT,0)                                      EVA   97
000332          EXP=STORLT(TERM,IFREE(C))                                   EVA   98
000336          CALL DELETA(C,0)                                            EVA   99
000340          ST1=IFREE(C)                                                EVA  100
000342          IFREE(C)=STORD(ST1,ST2)                                     EVA  101
000345          CALL DELETA(ST1)                                            EVA  102
000346          CALL DELETA(ST2)                                            EVA  103
000352          GO TO 200                                                   EVA  104
000354    1100  IF(STORE(STORE-C)-N(46))1300,1105,1300                      EVA  105
000357    1105  KA=IFREE(C)                                                 EVA  106
000361          CALL DELETA(C,0)                                            EVA  107
000363          KB=IFREE(C)                                                 EVA  108
000365          CALL ARY(KA,IGEN,KC,0)                                      EVA  109
000370    1104  IF(KB)1101,1101,1102                                        EVA  110
000373    1101  CALL ERR2(30,KC)                                            EVA  111
000375          IF(KB)1200,1200,1103                                        EVA  112
```

```
                1103  CALL ANY(IFREE(KC+1),ICOM,KC,0)                    EVA      113
                      GO TO 1104                                         EVA      114
                1102  CALL ANY(KB,IPAT15,J,0)                            EVA      115
                      IF(PATL.EQ.1)GO TO 1200                            COR2      18
                      KB=CELL(KB)                                        EVA      117
                      KB=APND(KB)                                        EVA      118
                      IFREE(C)=SHIFT(KB)                                 EVA      119
                1200  CALL CONCAT(IFREE(C),IFREE(C),ICOM)                EVA      120
                      CALL ANY(KA,IPAT15,J,0)                            EVA      121
                      IF(PATL.EQ.1)GO TO 667                             COR2      19
                      KA=CELL(KA)                                        EVA      123
                      KA=APND(KA)                                        EVA      124
                      KA=SHIFT(KA)                                       COR2      20
                667   CALL CONCAT(IFREE(C),IFREE(C),KA)                  COR2      21
                      CALL DELETE(KA)                                    COR8     168
                      GO TO 200                                          EVA      128
                1300  IF(IFREE(SUBJC)-9(48))1400,1301,1400               EVA      133
                1301  KA=IFREE(C)                                        EVA      134
                      KC=KA                                              EVA      135
                1304  CALL ANY(KA,ICOM,KB,0)                             EVA      136
                      IF(PATL.EQ.1)GO TO 1302                            EVA      137
                      CALL CAR2(KA,KB)                                   EVA      138
                      IF(PATL.EQ.0)GO TO 1303                            EVA      139
                      KC=IFREE(KA+1)                                     EVA      140
                      GO TO 1304                                         EVA      141
                1302  CALL ANY(KA,IPAT15,J,0)                            COR2      22
                      IF(PATL.EQ.1)GO TO 1303                            COR2      23
                      KA=CDR(KA)                                         COR2      24
                      KA=CELL(KA)                                        EVA      143
                      KA=APND(KA)                                        EVA      144
                1303  CALL DELETE(C,0)                                   EVA      145
                      CALL CONCAT(IFREE(C),IFREE(C),ILP)                 EVA      146
                      CALL CONCAT(IFREE(C),IFREE(C),KA)                  EVA      147
                      CALL CONCAT(IFREE(C),IFREE(C),IRP)                 EVA      148
                      CALL DELETE(KA)                                    EVA      149
                      GO TO 200                                          EVA      150
                1400  CALL ANY(SUBJC,IPAT10,KA,1)                        EVA      151
                      IF(PATL)1401,1401,1700                             EVA      152
                1401  CALL SPAN(IFREE(C),IPAT11,KA,KB,1)                 EVA      153
                      IF(PATL)1500,1404,1403                             EVA      154
                1404  IF(IFREE(KB+1))1403,1500,1403                      EVA      155
                1403  CALL ANY(IFREE(C),IPAT15,KB,0)                     EVA      156
                      IF(PATL.EQ.1)GO TO 1500                            EVA      157
                      SUBJC=CELL(IFREE(C))                               EVA      158
                      KA=APND(KB)                                        EVA      159
                      IF(KA(C)=SUBJC)                                    EVA      160
                1500  KA=IFREE(C)                                        EVA      161
                      CALL DELETE(C,0)                                   EVA      162
                      CALL SPAN(IFREE(C),IPAT11,KB,KC,1)                 EVA      163
                      IF(PATL)1504,1504,1503                             EVA      164
                1504  IF(IFREE(KC+1))1503,1600,1503                      EVA      165
                1503  CALL ANY(IFREE(C),IPAT15,J,0)                      EVA      166
                      IF(PATL.EQ.1)GO TO 1600                            EVA      167
                      KB=CELL(IFREE(C))                                  EVA      168
```

```
000630         KQ=[illegible](K[illegible])                                        EVA    169
000632         IFR[illegible](C)=S[illegible]T(KQ)                                 EVA    170
000636   1600  CALL IDENT(IFREE(C),KA)                                             EVA    171
000641         IF(FAIL.EQ.1)GO TO 1601                                             EVA    172
000643         IF(IFREE(SUBJEC).EQ.Q(44).OR.IFREE(SUBJEC).EQ.Q(60).OR.             EVA    173
             C  IFREE(SUBJEC)       .EQ.Q(61))GO TO 1602                           EVA    174
000657         CALL DELETE(KA)                                                     EVA    175
000661         CALL COPY(IFREE(C),IFALSE,0)                                        EVA    176
000665         GO TO 200                                                           EVA    177
000667   1602  CALL DELETE(KA)                                                     EVA    178
000671         CALL COPY(IFREE(C),ITRUE,0)                                         EVA    179
000675         GO TO 200                                                           EVA    180
000677   1601  IF(IFREE(SUBJEC).EQ.Q(59))GO TO 1603                                EVA    181
000702         IF(IFREE(SUBJEC).EQ.Q(61))GO TO 1604                                EVA    182
000704         IF(IFREE(SUBJEC).EQ.Q(44).AND.IFREE(KA+1).EQ.0.AND.IFREE(KA).EQ.    EVA    183
             1 Q(27))GO TO 1605                                                    EVA    184
000716         GO TO 1605                                                          EVA    185
000716   1603  IFREE(SUBJEC)=Q(58)                                                 EVA    186
000720         GO TO 1605                                                          EVA    187
000721   1604  IFREE(SUBJEC)=Q(60)                                                 EVA    188
000723   1606  IT[illegible]=IFREE(C)                                              EVA    189
000725         IFREE(C)=KA                                                         EVA    190
000727         KA=IT[illegible]                                                    EVA    191
000730   1605  IPA=IFREE(C)                                                        EVA    192
000732         IF(IFREE(IPA).EQ.Q(27).AND.IFREE(IPA+1 ).EQ.0)GO TO 1607            EVA    193
000742         CALL COPY(I[illegible],[illegible],0)                               EVA    194
000746         ST1=S[illegible](I[illegible],IFREE(C))                             EVA    195
000747         ST2=KA                                                              EVA    196
000751         KA=STA[illegible](ST1,ST2)                                          EVA    197
000753         CALL DELETE(ST1)                                                    EVA    198
000755         CALL DELETE(ST2)                                                    EVA    199
000757         CALL INSERT(IFREE(C),0,Q(27 ))                                      EVA    200
000763         KA=[illegible](KA)                                                  EVA    201
000765         KA=[illegible](KA)                                                  EVA    202
000767         KA=[illegible](KA)                                                  EVA    203
000772   1607  CALL [illegible](I[illegible](C),IFREE(C),SUBJEC)                   EVA    204
000776         CALL [illegible](IFREE(C),IFREE(C),KA)                              EVA    205
001002         IF(IFREE(SUBJEC).NE.Q(58).AND.IFREE(SUBJEC).NE.Q(60))GO TO 1608     CORRQ  105
001011         CALL SPAN(KA,DIGITS,KZ,KE,1)                                        CORRQ  106
001015         IF(FAIL.EQ.1)GO TO 1608                                             CORRQ  107
001021         IF(IFREE(KZ+1).NE.0)GO TO 1608                                      CORRQ  108
001023         CALL DELETE(KA)                                                     CORRQ  109
001025         CALL COPY(IFREE(C),ITRUE,0)                                         CORRQ  110
001030         GO TO 200                                                           CORRQ  111
001032   1608  CALL DELETE(KA)                                                     CORRQ  112
001034         GO TO 200                                                           EVA    207
001036   1700  IF(IFREE(SUBJEC)-Q(62))1700,1701,1900                               EVA    208
001041   1701  CALL IDENT(IFREE(C),ITRUE)                                          EVA    209
001044         IF(FAIL.EQ.1)GO TO 1702                                             EVA    210
001047         CALL COPY(IFREE(C),IFALSE,0)                                        EVA    211
001053         GO TO 200                                                           EVA    212
001055   1702  CALL IDENT(IFREE(C),IFALSE)                                         EVA    213
001060         IF(FAIL.EQ.1)GO TO 1703                                             EVA    214
001063         CALL COPY(IFREE(C),ITRUE,0)                                         EVA    215
```

```
001067         GO TO 200                                      EVA   216
001071  1703   K8=IFNLS(C)                                    COR9    3
001073         K8=IFRFF(K8+1)                                 COR9    4
001075         CALL ANY(K8,IPATIO,KA,1)                       COR9    5
001100         IF(FAIL.EQ.1)GO TO 1704                        COR9    6
001103         IF(IFRFF(K8).NE.O(44))GO TO 1712               COR9    7
001106         IFRFF(K8)=O(52)                                COR9    8
001107         GO TO 200                                      COR9    9
001110  1712   IF(IFRFF(K8).NE.O(52))GO TO 1705               COR9   10
001113         IFRFF(K8)=O(44)                                COR9   11
001114         GO TO 200                                      COR9   12
001115  1705   IF(IFRFF(K8).NE.O(53))GO TO 1706               COR9   13
001120         IFRFF(K8)=O(60)                                COR9   14
001121         GO TO 1707                                     COR9   15
001122  1706   IFRFF(K8)=O(53)                                COR9   16
001124  1707   K8=IFRFF(K8+1)                                 COR9   17
001125         IF(IFRFF(K8).EQ.O(38))GO TO 1708               COR9   18
001131         CALL INSERT(K8,1,O(38))                        COR9   19
001134         K8=IFRFF(K8+1)                                 COR9   20
001136         K8=IFRFF(K8+1)                                 COR9   21
001137         GO TO 1709                                     COR9   22
001141  1708   CALL DELETA(K8,1)                              COR9   23
001143         K8=IFRFF(K8+1)                                 COR9   24
001145  1709   CALL ANY(K8,ANY,KA,0)                          COR9   25
001150         IF(FAIL.EQ.1)GO TO 200                         COR9   26
001153         IF(IFRFF(K8).EQ.O(37))GO TO 1710               COR9   27
001156         IFRFF(K8)=O(37)                                COR9   28
001157         GO TO 1711                                     COR9   29
001160  1710   IFRFF(K8)=O(33)                                COR9   30
001162  1711   K8=IFRFF(KA+1)                                 COR9   31
001164         GO TO 1709                                     COR9   32
001167  1704   CALL [illegible](IFRFF(C),IFRFF(C),IRP)        COR9   33
001171         CALL INSERT(IFRFF(C),0,O(41))                  EVA   218
001176  1900   CALL INSERT(IFRFF(C),0,O(52))                  EVA   219
001201         GO TO 200                                      EVA   220
001203  1900   IF(IFRFF(SUBJC)-O(55))2200,1901,2200           EVA   221
      C THE SUBJECT IS .AND.                                  EVA   222
      C PRINCIPLES OF LOGIC USED ARE    ...  A .AND. .FALSE.   IFF .FALSE   EVA   223
      C                                 ... A .AND. .TRUE. IFF A            EVA   224
      C IF THE TOP OF THE STACK IS NOT .FALSE. GO TO 1902     EVA   225
001205  1901   CALL IDENT(IFRFF(C),IFALSE)                    EVA   226
001211         IF(FAIL.EQ.1)GO TO 1902                        EVA   227
      C IF IT IS , POP THE STACK AND CHANGE THE TOP OF THE STACK TO .FALSE.   EVA   228
001214         CALL DELETA(IFRFF(C))                          EVA   229
001217         CALL DELETA(C,0)                               EVA   230
001221         CALL COPY(IFRFF(C),IFALSE,0)                   EVA   231
001225         GO TO 200                                      EVA   232
001227  1902   CALL IDENT(IFRFF(C),ITRUE)                     EVA   233
      C IF SUBJECT IS THE .TRUE. GO TO 2000                   EVA   234
001232         IF(FAIL.EQ.1)GO TO 2000                        EVA   235
001235         CALL DELETA(IFRFF(C))                          EVA   236
001240         CALL DELETA(C,0)                               EVA   237
001242         GO TO 200                                      EVA   238
001244  2000   KA=IFRFF(C)                                    EVA   239
```

```
001246          CALL DELETA(C,0)                                        EVA     240
        C ASSIGN KA THE VALUE AT THE TOP OF THE STACK                   EVA     241
        C POP THE STACK                                                 EVA     242
        C IF THE VALUE OF THE TOP OF THE STACK IS NOT .FALSE. GO TO 2001 EVA    243
        C IF IT IS,DELETE KA                                            EVA     244
001250          CALL IDENT(IFREE(C),IFALSE)                             EVA     245
001253          IF(FAIL.EQ.1)GO TO 2001                                 EVA     246
001255          CALL DELETB(KA)                                         EVA     247
001260          GO TO 200                                               EVA     248
001262    2001  CALL IDENT(IFREE(C),ITRUE)                              EVA     249
        C IF THE VALUE OF THE TOP OF THE STACK IS NOT .TRUE. GO TO 2002 EVA     250
        C OTHERWISE POP THE STACK                                       EVA     251
        C AND PUSH ON KA                                                EVA     252
001265          IF(FAIL.EQ.1)GO TO 2002                                 EVA     253
001270          CALL DELETB(IFREE(C))                                   EVA     254
001273          IFREE(C)=KA                                             EVA     255
001275          KA=1                                                    EVA     256
001276          GO TO 200                                               EVA     257
        C HERE NEITHER OF THE OPERANDS ARE LOGICAL CONSTANTS            EVA     258
001277    2002  KTOP=IFREE(C)                                           EVCOR     1
001301    2003  CALL ANY(KTOP,I350,KZ,1)                                EVCOR     2
001304          IF(FAIL.EQ.1)GO TO 2004                                 EVA     260
001307          CALL FAL2(IFREE(C),KZ)                                  EVA     261
001312          IF(FAIL.EQ.0)GO TO 2100                                 EVA     262
001314          KTOP=IFREE(KZ+1)                                        EVCOR     3
001316          GO TO 2003                                              EVCOR     4
        C IF TEXT IS ALREADY .AND. IN THE TOP OF THE STACK SURROUND BY BALANCED EVA 263
        C EXPRESSIONS OR THE TOP OF  THE STACK IS A BALANCED ESPRESSION SURROUND EVA 264
        C BY PARENTHESES THEN LEAVE IT ALONE OTHERWISE THE TOP OF THE STACK EVA 265
        C BECOMES (#(TOP OF STACK))#                                    EVA     266
001317    2004  KCC=ISTAT(IFREE(C))                                     EVA     267
001323          CALL LT(IFREE(C),KCC,KZ)                                EVA     268
001326          KCC=IFREE(C)                                            EVA     269
001330          IF(IFREE(KZ).NE.Q(42).OR.IFREE(KCC).NE.Q(41))GO TO 2005 EVA     270
001341          CALL FAL2(IFREE(KCC+1),KZ)                              EVA     271
001344          IF(FAIL.EQ.0)GO TO 2100                                 EVA     272
001346    2005  CALL CONCAT(IFREE(C),IFREE(C),IRP)                      EVA     273
001352          CALL INSERT(IFREE(C),0,Q(41))                           EVA     274
        C KA BECOMES #(A KA #)#                                         EVA     275
        C UNLESS KA IS ALREADY   AIN THE TOP OF THE STACK, BALANCED     EVA     276
        C EXPRESSIONS ON EITHER SIDE OF IT, IF IT IS LEAVE TOP OF STACK ALONE EVA 277
001356    2100  CALL CONCAT(KA,KA,IRP)                                  EVA     278
001361          CALL INSERT(KA,0,Q(41))                                 EVA     279
001364          CALL IDENT(KA,IFREE(C))                                 EVA     280
001367          IF(FAIL.EQ.0)GO TO 2101                                 EVA     281
001371          KE=IFREE(C)                                             EVA     282
001373          KCC=IFREE(C)                                            EVA     283
001375    2104  CALL ANY(KE,I350,KE,0)                                  EVA     284
001400          IF(FAIL.EQ.1)GO TO 2102                                 EVA     285
001403          CALL FAL2(KCC,KE)                                       EVA     286
001405          IF(FAIL.EQ.0)GO TO 2103                                 EVA     287
001407          KE=IFREE(KE+1)                                          EVA     288
001411          GO TO 2104                                              EVA     289
001412    2105  CALL COPY(KD,KCC,KE)                                    EVA     290
```

```
001415            CALL INSRT(KQ,KA)                                    EVA   291
001417            CALL DELETS(KQ)                                      EVA   292
001421            IF(FAIL.EQ.0)GO TO 2101                              EVA   293
001423            KCC=IFREE(K+1)                                       EVA   294
001425            KZ=KCC                                               EVA   295
001427            GO TO 2104                                           EVA   296
001427   2102     CALL COPY(K),KCC,0)                                  EVA   297
001432            CALL INSRT(KQ,KA)                                    EVA   298
001434            CALL DELETS(KQ)                                      EVA   299
001436            IF(FAIL.EQ.0)GO TO 2101                              EVA   300
001440            CALL INSERT(K+1,0,0(55))                             EVA   301
          C IF 001, TOP OF STACK RECOMES TOP OF STACK .AND. KA                 302
001443            CALL CONCAT(IFREE(C),IFREE(C),KA)                    EVA   303
001447   2101     CALL DELETS(KA)                                      EVA   304
001451            GO TO 203                                            EVA   305
001453   2200     IF(IFREE(STACK(C)-0(54))2500,2201,2500               EVA   306
001455   2201     CALL INSERT(IFREE(C),IFALSE)                         EVA   307
001461            IF(FAIL.EQ.1)GO TO 2202                              EVA   308
001464            CALL DELETS(IFREE(C))                                EVA   309
001467            CALL DELETS(C,0)                                     EVA   310
001471            GO TO 203                                            EVA   311
001473   2202     CALL INSERT(IFREE(C),ITRUE)                          EVA   312
001477            IF(FAIL.EQ.1)GO TO 2203                              EVA   313
001501            CALL DELETS(IFREE(C))                                EVA   314
001504            CALL DELETS(C,0)                                     EVA   315
001511            CALL COPY(IFREE(C),ITRUE,0)                          EVA   316
001512            GO TO 203                                            EVA   317
001514   2203     CALL CONCAT(IFREE(C),IFREE(C),IRP)                   EVA   318
001521            CALL INSERT(IFREE(C),0,0(41))                        EVA   319
001524   2300     KA=IFREE(C)                                          EVA   320
001526            CALL DELETS(C,0)                                     EVA   321
001531   2301     CALL INSERT(IFREE(C),IFALSE)                         EVA   322
001535            IF(FAIL.EQ.1)GO TO 2302                              EVA   323
001537            CALL DELETS(IFREE(C))                                EVA   324
001541              IFREE(C)=KA                                        EVA   325
001543            KA=1                                                 EVA   326
001544            GO TO 203                                            EVA   327
001546   2302     CALL INSERT(IFREE(C),ITRUE)                          EVA   328
001550            IF(FAIL.EQ.1)GO TO 2303                              EVA   329
001553            CALL DELETS(KA)                                      EVA   330
001555            GO TO 203                                            EVA   331
001557   2303     CALL CONCAT(IFREE(C),IFREE(C),IRP)                   EVA   332
001563            CALL INSERT(IFREE(C),0,0(41))                        EVA   333
001567   2400     CALL INSERT(KA),0,0(54))                             EVA   334
001572            CALL CONCAT(IFREE(C),IFREE(C),KA)                    EVA   335
001576            CALL DELETS(KA)                                      EVA   336
001600            GO TO 203                                            EVA   337
001602   2500     CALL INSERT(IFREE(C),IFALSE)                         EVA   338
001605            IF(FAIL.EQ.1)GO TO 2502                              EVA   339
001610            KA=IFREE(C)                                          EVA   340
001612            CALL DELETS(C,0)                                     EVA   341
001614            CALL INSERT(IFREE(C),ITRUE)                          EVA   342
001617            IF(FAIL.EQ.1)GO TO 2503                              EVA   343
001622            CALL DELETS(KA)                                      EVA   344
```

```
001624          CALL COPY(IFREF(C),IFALSE,0)                          EVA     345
001630          GO TO 200                                             EVA     346
001633    2503  CALL FORMIT(IFREF(C),IFALSE)                          EVA     347
001635          IF(FAIL.EQ.1)GO TO 2506                               COR6A     1
001640          CALL DELETS(KA)                                       EVA     349
001642          CALL COPY(IFREF(C),ITRUE)                             EVA     350
001646          GO TO 200                                             EVA     351
001647    2502  CALL FORMIT(IFREF(C),ITRUE)                           EVA     352
001652          IF(FAIL.EQ.1) GO TO 2505                              EVA     353
001655          CALL DELETS(IFREF(C))                                 EVA     354
001660          CALL DELETA(C,0)                                      EVA     355
001662          CALL COPY(IFREF(C),ITRUE,0)                           EVA     356
001666          GO TO 200                                             EVA     357
        C  (EXP .IMPLIES. .FALSE.)IS .NOT.EXP                         COR6A     2
001671    2504  CALL DELETS(KA)                                       COR6A     3
001672          CALL CONCAT(IFREF(C),IFREF(C),IRP)                    COR6A     4
001675          CALL INSERT(IFREF(C),0,0(41))                         COR6A     5
001703          CALL INSERT(IFREF(C),0,0(42))                         COR6A     6
001706          GO TO 200                                             COR6A     7
001711    2505  KA=IFREF(C)                                           COR2     26
001712          CALL DELETA(C,0)                                      EVA     361
        C  (.TRUE. .IMPLIES. EXP) IS EXP                              COR6A     8
001714          CALL FORMIT(IFREF(C),ITRUE)                           COR6A     9
001717          IF(FAIL.EQ.1)GO TO 2507                               COR6A    10
001722          CALL DELETA(IFREF(C))                                 COR6A    11
001725          IFREF(C)=KA                                           COR6A    12
001727          KA=0                                                  COR6A    13
001730          GO TO 200                                             COR6A    14
        C  (.FALSE. .IMPLIES. EXP) IS .TRUE.                          COR6A    15
001731    2507  CALL FORMIT(IFREF(C),IFALSE)                          COR6A    16
001734          IF(FAIL.EQ.0)GO TO 2600                               COR6A    17
001737          CALL DELETS(KA)                                       COR6A    18
001741          CALL COPY(IFREF(C),ITRUE,0)                           COR6A    19
001745          GO TO 200                                             COR6A    20
001747    2600  CALL SAV(KA,ILP,KCC,1)                                COR2     27
001753          IF(FAIL.EQ.0)GO TO 2617                               EVA     363
001756          CALL CONCAT(KA,KA,IRP)                                EVA     364
001763          CALL INSERT(KA,0,0(41))                               EVA     365
001767    2617  CALL SAV(IFREF(C),ILP,KCC,1)                          EVA     366
001772          IF (FAIL .EQ. 0) GO TO 2616                           EVA     367
001773          CALL CONCAT(IFREF(C),IFREF(C),IRP)                    EVA     368
001776          CALL INSERT(IFREF(C),0,0(41))                         EVA     369
002000    2616  LISTL=1                                               COR7     1
002001          LISTLN=0                                              COR7     2
002002          CALL SEPAT(KA,LISTL,LISTLN,IAND,IPAT10)               COR7     3
002010          IF(LISTLN.EQ.0)GO TO 2620                             COR7     4
002013          LISTLN=1                                              COR7     5
002015          LISTL=0                                               COR7     6
002016          CALL SEPAT(IFREF(C),LISTL,LISTLN,IAND,IPAT10)         COR7     7
002024          IF(LISTL.EQ.0)GO TO 2621                              COR7     8
002027          CALL DELETS(KA)                                       COR7     9
002031          CALL DELETA(LISTLN)                                   COR7    10
002033    2623  I=EL=IFREF(LISTL)                                     COR7    11
002037          CALL DELETA(LISTL,0)                                  COR7    12
```

```
002027          CALL SETK(IPFL,LISTL)                                   COR7    13
002021          IF(IAIL.EQ.0)GO TO 2622                                 COR7    14
002033          IF(LISTN.EQ.0)GO TO 2627                                COR7    15
002034          CALL CONCAT(LISTPN,LISTPN,IAN))                         COR7    16
002037          CALL CONCAT(LISTPN,LISTPN,IREL)                         COR7    17
002042    2622  CALL DELETA(IPFL)                                       COR7    18
002044          GO TO 2628                                              COR7    19
002047    2627  LISTPN=IREL                                             COR7    20
002050    2628  IF(LISTP.GT.0)GO TO 2623                                COR7    21
002053          IF(LISTPN.EQ.0)GO TO 2624                               COR7    22
002054          KA=LISTPN                                               COR7    23
002055    2625  IF(LISTP.EQ.0.0)GO TO 2626                              COR7    24
002056          KA=IFREE(LISTL)                                         COR7    25
002060          CALL DELETA(KA)                                         COR7    26
002061          CALL DELETA(LISTL,0)                                    COR7    27
002063          GO TO 2625                                              COR7    28
002065    2624  CALL COPY(LEXPRE(C),ITRUE,0)                            COR7    29
002071          GO TO 200                                               COR7    30
002073    2621  IF(LISTP.EQ.0)GO TO 2626                                COR7    31
002074          KA=IFREE(LISTL)                                         COR7    32
002076          CALL DELETA(KA)                                         COR7    33
002077          CALL DELETA(LISTP,0)                                    COR7    34
002101          GO TO 2621                                              COR7    35
002103    2623  CALL DELETA(LISTPN)                                     COR7    36
002105    2623  CALL CONCAT(KA,KA,IRP)                                  COR7    37
002111          CALL INSERT(KA,0,0(41))                                 EVA     439
002113          CALL INSERT(KA,0,0(53))                                 EVA     440
002117          CALL INSERT(KA,0,0(42))                                 EVA     441
002121          CALL INSERT(IPRE(C),IPRE(C),KA)                         EVA     442
002125          CALL INSERT(IPRE(C),0,0(41))                            EVA     443
002131          CALL DELETA(KA)                                         EVA     444
002133          GO TO 200                                               EVA     445
002135    2700  KA=IFREE(C)                                             EVA     446
002137          CALL COPY(KA,ITRUE)                                     EVA     447
002141          IF(IAIL.EQ.0)GO TO 2701                                 EVA     448
002144          IF(IPRE(KA)=0(5)                                        COR6    23
002150          CALL INSERT(KA,0,0(14))                                 COR6    24
002151          CALL INSERT(KA,0,0(15))                                 COR6    25
002154          CALL INSERT(KA,0,0(14))                                 COR6    26
002157          KA,L=0                                                  COR8P   199
002160          GO TO 2702                                              EVA     459
002162    2701  CALL COPY(KA,IFALSE)                                    EVA     460
002164          IF(IAIL.EQ.0)GO TO 2703                                 COR8A   8
002167          IF(IPRE(KA)=0(5))                                       COR6    27
002171          CALL INSERT(KA,0,0(14))                                 COR6    28
002174          CALL INSERT(KA,0,0(5))                                  COR6    29
002177          CALL INSERT(KA,0,0(20))                                 COR6    30
002202          CALL INSERT(KA,0,0(19))                                 COR6    31
002205          CALL INSERT(KA,0,0(9))                                  COR6    32
002210          CALL INSERT(KA,0,0(19))                                 COR6    33
002213          CALL INSERT(KA,0,0(14))                                 EVA     476
002215          CALL INSERT(KA,0,0(15))                                 EVA     477
002221          CALL INSERT(KA,0,0( 3))                                 EVA     478
002224          CALL INSERT(KA,0,0(45))                                 EVA     479
```

```
002227          CALL INSERT(EVA,0,0(20))                    EVA     480
002232          CALL INSERT(EVA,0,0(15))                    EVA     481
002236          CALL INSERT(EVA,0,0(14))                    EVA     482
002241          CALL INSERT(EVA,0,0(45))                    EVA     483
002245          CALL INSERT(EVA,0,0(13))                    EVA     484
002250          CALL INSERT(EVA,0,0( 9))                    EVA     485
002255          CALL INSERT(EVA,0,0(45))                    EVA     486
002261          CALL INSERT(EVA,0,0( 8))                    EVA     487
002264          CALL INSERT(EVA,0,0(20))                    EVA     488
002267          CALL INSERT(EVA,0,0( 1))                    EVA     489
002272          CALL INSERT(EVA,0,0(16))                    EVA     490
002275          CALL INSERT(EVA,0,0(45))                    EVA     491
002277          CALL INSERT(EVA,0,0(19))                    EVA     492
002303          CALL INSERT(EVA,0,0( 9))                    EVA     493
002306          CALL INSERT(EVA,0,0( 5))                    EVA     494
002311          CALL INSERT(EVA,0,0(20))                    EVA     495
002314          FAIL=1                                      COR8P   200
002317  2702      CALL DELETE(ITRUE)                        EVA     496
002321          CALL DELETE(IFALSE)                         EVA     497
002324          CALL DELETE(IWAK)                           EVA     498
002327          CALL DELETE(IVAL)                           EVA     499
002332          CALL DELETE(ICOM)                           EVA     500
002335          CALL DELETE(IOP)                            EVA     501
002340          CALL DELETE(ILP)                            EVA     502
002343          CALL DELETE(IPAT10)                         EVA     503
002347          CALL DELETE(IALPH)                          EVA     504
002352          CALL DELETE(IPAT11)                         EVA     505
002355          CALL DELETE (IA13)                          EVA     506
002360          CALL DELETE(IPAT15)                         EVA     507
002363          CALL DELETE(SAVY)                           COR9    34
002367          CALL DELETE(C)                              EVA     508
002372          GET EVA                                     EVA     509
002376  2703    IF(IPAT11.EQ.EVA(41))GO TO 2705             COR8P   201
002402          CALL CONCAT(IPAT11,IPAT11,IPAT15)           COR8A   10
002407          CALL SUB(EVA,IPAT11,KA,K8,1)                COR8A   11
002414          IF(FAIL.EQ.1) GO TO 2705                    COR8P   202
002420          IF(EVA(KA+1).GT.0)GO TO 2705                COR8P   203
002425          CALL ADD(EVA,IOP,KA,0)                      COR8A   14
002430          IF(FAIL.EQ.1)GO TO 2704                     COR8A   15
002434          CALL CAR(EVA,KA)                            COR8A   16
002440          IF(FAIL.EQ.0)GO TO 2705                     COR8P   204
002443  2704    EVA=GOUTC(EVA)                              COR8A   18
002451          EVA=LOWER(EVA)                              COR8A   19
002455          EVA=SHORT(EVA)                              COR8A   20
002461  2705    FAIL=-1                                     COR8P   205
002465          GO TO 2702                                  COR8A   21
002471          END                                         EVA     510
```

```
                SUBROUTINE DIFF(IREL,LOOK,ISIGN,IOP)                                DIFF     2
C00007          COMMON/BLOCK/SPACE(14000),FAIL,FREE/DIGOP/IDIGS,I1,I2,I3,I4         BIGGER   22
C00007          DIMENSION SPACE,FAIL,FREE,OPCODE                                    DIFF     4
C00007          DATA LT/2HLT/,L=/2HL=/,IEQ/1H=/,IMINUS/1H-/,IPLUS/1H+/              COR8R    35
C00007          IF(LOOK.NE.0) GO TO 1                                               DIFF     6
C00010          CALL LEX(IREL,K,K1)                                                 DIFF     7
C00012          IF(FAIL.EQ.1)RETURN                                                 DIFF     8
C00017          ISIGN=0                                                             DIFF     9
C00020          K2=K1                                                               DIFF     10
C00021          K=K1                                                                DIFF     11
C00022          IF(SPACE(K1).NE.IMINUS)GO TO 2                                      DIFF     12
C00025          ISIGN = 1                                                           DIFF     13
C00026          K2=SPACE(K1+1)                                                      DIFF     14
C00027        2 CALL SPAN(K2,IDIGS,KA,KB,L)                                         DIFF     15
C00033          IF(FAIL.NE.0)RETURN                                                 DIFF     16
C00040          JUMP=0                                                              DIFF     17
C00041          CALL COPY(IDIGS,KA,KB)                                              DIFF     18
C00044          JUMP=DECODE(JUMP)                                                   DIFF     19
C00046          IF(ISIGN.EQ.1) JUMP=-JUMP                                           DIFF     20
C00054          CALL DELETE(IDIGS)                                                  DIFF     21
C00056          K1=SPACE(K1+1)                                                      DIFF     22
C00063          IF(SPACE(K1).EQ.IPLUS)K1=SPACE(K1+1)                                COR8R    34
C00066          GOTO(3                                                              DIFF     23
C00067        1 CALL LEX(LOOK,3,K)                                                  DIFF     24
C00072          IOP=SPACE(K)                                                        DIFF     25
C00074          IF(IOP.LT.LT.OR.IOP.EQ.LE.OR.IOP.EQ.IEQ) GO TO 3                    DIFF     26
C00110        4 ISIGN = 2                                                           DIFF     27
C00111          GOTO 5                                                              DIFF     28
C00112        3 ISIGN=0                                                             DIFF     29
C00113          JUMP=0                                                              DIFF     30
C00114          JUMP=0                                                              DIFF     31
C00114          K = SPACE(K+1)                                                      DIFF     32
C00116          IF(K.EQ.0) GO TO 4                                                  DIFF     33
C00117          K2=K                                                                DIFF     34
C00117          IF(SPACE(K).NE.IMINUS)GO TO 5                                       DIFF     35
C00122          ISIGN=1                                                             DIFF     36
C00122          K2 = SPACE(K+1)                                                     DIFF     37
C00124        5 CALL SPAN(K2,IDIGS,KA,KB,L)                                         DIFF     38
C00130          IF(FAIL.NE.0)GO TO 60                                               COR8D    113
C00135          K=SPACE(K+1)                                                        DIFF     40
C00136          IF(SPACE(K).EQ.IPLUS)K=SPACE(K+1)                                   COR8R    37
C00142          CALL LEX(IREL,K)                                                    DIFF     41
C00144          IF(FAIL.NE.0)GO TO 4                                                DIFF     42
C00151          CALL COPY(IDIGS,KA,KB)                                              DIFF     43
C00153          JUMP=DECODE(IDIGS)                                                  DIFF     44
C00156          CALL DELETE(IDIGS)                                                  DIFF     45
C00157          IF(ISIGN.NE.1) JUMP=-JUMP                                           DIFF     46
C00165        6 ISIGN=JUMP-JUMP                                                     DIFF     47
C00167          IF(ISIGN.GT.0)GO TO 10                                              DIFF     48
C00172          IF(ISIGN.LT.-1)GO TO 20                                             DIFF     49
C00174          IF(ISIGN.EQ.0)GO TO 7                                               COR8A    22
C00175          ISIGN=-1                                                            COR8A    23
C00176          GOTO                                                                DIFF     51
```

```
000176    70    CALL IDENT(K1,K)
000200          IF(FAIL.EQ.1)GO TO 4
000205          ISIGN=0
000205          RETURN
000206    10    ISIGN=1
000207          RETURN
000210    20    ISIGN=-2
000211          RETURN
000212    60    CALL IDENT(K1,K)
000214          IF(FAIL.EQ.1)GO TO 4
000221          GO TO 6
000221          END
```

```
CDR8A     24
CDR8A     25
CDR8A     26
CDR8A     27
DIFF      52
DIFF      53
DIFF      54
DIFF      55
CDR8Q    114
CDR8Q    115
CDR8Q    116
DIFF      56
```

```
                      SUBROUTINE SEPRAT(KA,KLIST,NOLIST,IAND,IPATIO)                SEPRAT     2
000010                COMMON/BLOCK/SPACE(14000),FAIL,FREE                           BIGGER    24
000010                INTEGER SPACE,FAIL,FREE                                       SEPRAT     4
                      C THIS SUBROUTINE TAKES CONJUNCTS OF THE EXPRESSION KA WITH RELATIONS  SEPRAT     5
                      C AND INSERTS THEM IN THE LIST KLIST AND ALL OTHERS ARE CONCATENATED   SEPRAT     6
                      C INTO LIST NOLIST                                            SEPRAT     7
000010                CALL ANY(KA,IPATIO,K,0)                                       SEPRAT     8
000012                IF(FAIL.EQ.1)RETURN                                           SEPRAT     9
000020                CALL NLIST(NOLIST,L)                                          SEPRAT    10
000022                K1=K                                                          SEPRAT    11
000024          2623  CALL ANY(K1,IAND,K,0)                                         SEPRAT    12
000031                IF(FAIL.EQ.1)GO TO 2620                                       SEPRAT    13
000036          2624  CALL SAL2(KA,K)                                               SEPRAT    14
000040                IF(FAIL.EQ.1)GO TO 2621                                       SEPRAT    15
000045                CALL COPY(K1,K3,K)                                            SEPRAT    16
000050          2625  CALL ANY(K2,IPATIO,K1,0)                                      SEPRAT    17
000054                IF(FAIL.EQ.1)GO TO 2622                                       SEPRAT    18
000060                CALL LINSERT(NLIST,J,K2)                                      SEPRAT    19
000064          2627  IF(K1.EQ.0)GO TO 2629                                         SEPRAT    20
000071                K=SPACE(K+1)                                                  SEPRAT    21
000073                GO TO 2623                                                    SEPRAT    22
000075          2622  CALL CONCAT(NLIST,NOLIST,IAND)                                SEPRAT    23
000075                CALL CONCAT(NLIST,NOLIST,K2)                                  SEPRAT    24
000102                CALL DELETA(K2)                                               SEPRAT    25
000104                GO TO 2624                                                    SEPRAT    26
000110          2621  CALL ANY(SPACE(K+1),IAND,K1,0)                                SEPRAT    27
000114                IF(FAIL.EQ.1)GO TO 2620                                       SEPRAT    28
000121                K=K1                                                          SEPRAT    29
000123                GO TO 2624                                                    SEPRAT    30
000125          2620  CALL COPY(K3,KA,0)                                            SEPRAT    31
000132                K=1                                                           SEPRAT    32
000137                GO TO 2626                                                    SEPRAT    33
000141          2629  CALL DELETA(NOLIST,0)                                         SEPRAT    34
000145                IF(NLIST.GT.0)CALL DELETA(NOLIST,0)                           SEPRAT    35
000147                RETURN                                                        SEPRAT    36
000151                END                                                           SEPRAT    37
```

```
                SUBROUTINE REDUCE(SLOCST)                                     REDUCE     2
000003          COMMON/BLOCK/SPACE(14000),FAIL,FREE/DIGOP/IDIG,IOP,ICOM,SEMICO,  BIGGER    27
               1ICON/ALPHA/Q(63)   /CON/COND                                  REDUCE     4
000003          INTEGER SLOCST,COND,SEMICO,FAIL,FREE,SPACE,PREV  ,Q           REDUCE     5
000003          CALL [illegible](SLOCST)                                      REDUCE     6
000004          IFLAGS=0                                                      REDUCE     7
000005          ISLO=SLOCST                                                   REDUCE     8
000007    100   CALL STRING(ISLO,COND,KA,KB,1)                                REDUCE     9
000013          IF(FAIL.EQ.1)GO TO 101                                        REDUCE    10
000015          ISLO=SPACE(KB+1)                                              REDUCE    11
000020          IF(IFLAGS.[illegible])GO TO 102                               REDUCE    12
000021          CALL [illegible](SLOCST,IFLAGS,KB)                            CORRED     2
000025    102   CALL ANY(ISLO,SEMICO,IFLAGS,0)                                REDUCE    18
000026          ISLO=SPACE(IFLAGS+1)                                          REDUCE    19
000031          GO TO 100                                                     REDUCE    20
000032    101   IF(ISLO.EQ.0)GO TO 103                                        CORRED     3
000033          CALL ANY(ISLO,SEMICO,KA,0)                                    REDUCE    22
000036          IF(SPACE(ISLO).EQ.Q(3))GO TO 105                              REDUCE    23
000042          IFLAGS=0                                                      REDUCE    24
000042          ISLO=SPACE(KA+1)                                              REDUCE    25
000044          GO TO 100                                                     REDUCE    26
000045    105   KA=SPACE(KA+1)                                                REDUCE    27
000047          CALL [illegible](SLOCST,ISLO,KA)                              REDUCE    28
000051          ISLO=KA                                                       REDUCE    29
000053          GO TO 100                                                     REDUCE    30
000054    103   CALL [illegible](ISC,Q(53))                                   CORRED     4
000057          CALL [illegible](ISC,0,Q(46))                                 CORRED     5
000062          ISC=SLOCST                                                    CORRED     6
000065    106   CALL [illegible](ISC,COND,KA,KB,1)                            CORRED     7
000070          IF(FAIL.EQ.1)GO TO 104                                        CORRED     8
000073    104   CALL [illegible](KA,ISC,KA,0)                                 CORRED     9
000076          IF(FAIL.EQ.1)GO TO 110                                        CORRED    10
000101          IF(SPACE(KA).[illegible].Q(63))GO TO 106                      CORRED    11
000104          CALL [illegible](ISC,KA)                                      CORRED    12
000106          IF(FAIL.EQ.1)GO TO 107                                        CORRED    13
000111          SPACE(KA)=Q(55)                                               CORRED    14
000113    107   KA=SPACE(KA+1)                                                CORRED    15
000115          GO TO 106                                                     CORRED    16
000117    10[illegible]  ISC=SPACE(KA+1)                                      CORRED    17
000120          GO TO 10[illegible]                                           CORRED    18
000121    10[illegible]  CALL [illegible](ISC,SEMICO,KA,0)                    CORRED    19
000124          IF(FAIL.EQ.1)GO TO 110                                        CORRED    20
000127          ISC=SPACE(KA+1)                                               CORRED    21
000131          GO TO 109                                                     CORRED    22
000132    110   CALL SELECT[illegible](ISC)                                   CORRED    23
000134          RETURN                                                        CORRED    24
000135          END                                                           REDUCE    31
```

SPECIAL ASSERTIONS AND FUNCTIONS FOR LIST COHERENCE

EXISTS(F(K), K, I, J)	is true if there exists K in the set (I..J) such that F(K) is true, and false otherwise.
EXSET(F(S), S, I, J)	is true if there exists a set S, which is a subset of (I..J), such that F(S) is true, and false otherwise.
UNIQUE(F(K), K, I, J)	The value of this function is that unique integer K in the set (I..J) such that F(K) is true. UNIQUE(F(K), K, I, J) has a value only if such a K exists and is unique (that is, there can be only one such K).
UNSET(F(S), S, I, J)	The value of this function is that unique set S, which is a subset of (I..J), such that F(S) is true. UNSET(F(S), S, I, J) has a value only if such a set exists and is unique.
EXFUNC(F(G), G, I, J)	is true if there exists a function G, whose domain is the set (I..J), such that F(G) is true, and false otherwise.
UNFUNC(F(G), G, I, J)	The value of this function is that unique function G, whose domain is (I..J), such that F(G) is true. UNFUNC(F(G), G, I, J) has a value only if such a function exists and is unique.
EVAL(F, N)	is F(N) (where F may, for example, be of the form UNFUNC(G(H), H, I, J)).
MEMBER(I, S)	is true if I is a member of the set S, and false otherwise.

SPECIAL ASSERTIONS AND FUNCTIONS FOR LIST COHERENCE (CONTINUED)

LISTLEN(SPACE, K) = UNIQUE(DEFUNC((K = G(1), ALL(SPACE(G(M)+1) = G(M+1), M, 1, N-1), SPACE(G(N)) = 0), G, 1, N), N, 1, 10000)

LISTLEN(SPACE, K) is the length of the list (in the list space SPACE) pointed to by K.

ADDR(SPACE, I, K) = EVAL(DEFUNC((K = G(1), ALL(SPACE(G(M)+1) = G(M+1), M, 1, LISTLEN(SPACE, K) - 1), SPACE(G(LISTLEN(SPACE, K))) = 0), G, 1, LISTLEN(SPACE, K)), I)

ADDR(SPACE, I, K) is the index of the I-th element of the list pointed to by K. ADDR(SPACE, 1, K) = K.

ELT(SPACE, I, K) = SPACE(ADDR(SPACE, I, K))

ELT(SPACE, I, K) is the I-th element of the list pointed to by K.

CARLIST(SPACE, K) = ELT(SPACE, 1, K)

CARLIST(SPACE, K) is the first element of the list pointed to by K.

CDRLIST(SPACE, K) = SPACE(ADDR(SPACE, 1, K) + 1)

CDRLIST(SPACE, K) points to a list obtained by removing the first element from the list to which K points.

MEMLIST(SPACE, X, K) = EXISTS(ADDR(SPACE, I, K) = X, I, 1, LISTLEN(SPACE, K))

MEMLIST(SPACE, X, K) is true if X is an index of an element in the list pointed to by K.

[illegible]

[illegible]

SICHL(SPACE, S) = EXISTS((MEMLIST(SPACE, I, H), MEMBER(H, S), .NOT. (EXISTS((MEMLIST(SPACE, I, G), MEMBER(G, S), G .NE. H), G, 1, 10000))), H, 1, 10000)

SICHL(SPACE, S) is true if S is a Set of Indices of Current Heads of Lists in a coherent sense; that is, every index H (in the list space SPACE) is on one and only one list starting at an element of S.

LIST.COH(SPACE) = EXIST((SICHL(SPACE, S), MEMBER(FREE, S)), S, 1, 10000)

LIST.COH(SPACE) says that SPACE is a coherent list space and that FREE points to its list of available space.

LIST.COH(SPACE, FREE) should be true throughout the operation of the SYMBOL simulator, except possibly within the basic list manipulation functions such as insertion, deletion, copying, getting a new cell, and so on. However, it is assumed to be true at the beginning of each of these, and, in turn, must be true at the end.

Note that we do not refer to lists as abstract entities. A list is denoted by the integer that points to its head. Thus there is no need for a "list value" (LV) function.

Three more functions are:

LIST.PTR(SPACE, X, K) = EXISTS(ADDR(I, K) = X, I, 1, LISTLEN(SPACE,K))

LIST.PTR(SPACE, X, K) is true if X is a pointer in range of the list pointed to by K.

LIST.IND(SPACE, X, K) = UNIQUE(ADDR(I, K) = X, I, 1, LISTLEN(SPACE,K))

LIST.IND(SPACE, X, K) is the index of X in the list pointed to by K. That is, if LIST.IND(SPACE, X, K) = J, then X is the J-th element in that list.

LIST.LSUB(SPACE, K1, K2) = LIST.IND(SPACE, K2, K1) - 1

LIST.LSUB(SPACE, K1, K2) is the length of the sublist (of some unnamed list) stretching from K1 to K3 where SPACE(K3+1) = K2.

This last function does some of the work that SUBLIST did before. There is no SUBLIST function any more.