```
            IDENT     START1
            TITLE                B C P L   C O M P I L E R -- PASS 1
            TITLE                     SYSTEM LINK
            SPACE     10
            LIST      -R
GLBSIZE     EQU       100
STKSIZE     EQU       9000
FRECAPS     EQU       10
            SPACE     5
            EXT       GLOBAL0
            ORG       0
            SPACE     5
            BSSZ      2
            DATA      L.BCPL.,L.CLASS.          CLASS CODE NAME
            DATA      3                      NO. MAP ENTRIES
            DATA      50                     SPACE FOR COMPILED MAP
            VFD       60/GLOBAL0+GLBSIZE+1+STKSIZE  FL
            VFD       60/ENTRY               ENTRY POINT
            VFD       60/NEXTCAP+FRECAPS
            VFD       60/GLBSIZE+STKSIZE+ENDCAPS    SCRH FILE SIZE
            TITLE                     LOGICAL MAP ENTRIES
            SPACE     5
            DATA      0,0,0,0                FOR SYS COMMUNICATION
            VFD       60/ENDCAPS,60/0        LENGTH, R/W
            SPACE     5
            DATA      L.BCPL.,L.S.             MAP ENTRY FOR CODE
            VFD       60/ENDCAPS           FILE ADDRESS
            VFD       60/ENDCAPS             CORE ADDRESS
            VFD       42/0,18/GLOBAL0-ENDCAPS+1      LENGTH
            DATA      1                      R/W
            SPACE     5
            DATA      0,0      SCRATCH FILE ENTRY
            VFD       60/ENDCAPS
            VFD       60/GLOBAL0
            VFD       60/GLBSIZE+STKSIZE      SIZE OF SC FILE
            DATA      0                      R/W
            SPACE     5
            DATA      -1                     END OF MAP SPECIFIERS
            TITLE                          CAPABILITIES
*
            SPACE     5
CAP14       DATA      L.CBLK.,L.OPERATE.
            DATA      L.PROBE.,L.OPERATE.
            DATA      L.DELBLK.,L.OPERATE.
SPRET       SET       *-CAP14
SPRET       SET       SPRET/2+14
            DATA      L.RETURN.,L.OPERATE.
            DATA      L.REDSHP.,L.OPERATE.
            DATA      L.DSPCLX.,L.OPERATE.
            DATA      L.FRETUR.,L.OPERATE.
            DATA      L.RESTOR.,L.OPERATE.
            DATA      L.UDAT.,L.OPERATE.
            DATA      L.CCLIST.,L.OPERATE.
            DATA      L.CSPROC.,L.OPERATE.
            DATA      L.CCC.,L.OPERATE.
            DATA      L.CAPOUT.,L.OPERATE.
```

```
          DATA      L.MKOPR.,L.OPERATE.
          DATA      L.DELCL.,L.OPERATE.
          DATA      L.MPCHRW.,L.OPERATE.
          DATA      L.MPCHRO.,L.OPERATE.
          DATA      L.DELSUB.,L.OPERATE.
          DATA      L.SAVE.,L.OPERATE.
          DATA      L.ALLOC.,L.OPERATE.          BCPL2 CLIST
          DATA      L.ALLOC.,L.OPERATE.          BCPL2 SCRATCH
          DATA      L.ALLOC.,L.OPERATE.          BCPL2 CLASS CODE
          DATA      L.ALLOC.,L.OPERATE.          BCPL2,S
          DATA      L.ALLOC.,L.OPERATE.          BCPL2 CALL OPR
ENDCAPS   DATA      0
NEXTCA    SET       ENDCAPS-CAP14
NEXTCAP   EQU       NEXTCA/2+14
          TITLE                  MACROS
          SPACE     5
*
*         MACROES TO LINK COMPASS WITH BCPL PROGRAMS
*
          SPACE     5
RESET     MACRO
          SB5       =XGLOBAL0
          SA2       B5
          SB6       X2
          SB1       1
          SB2       -1
          MX6       0
          ENDM
          SPACE     5
GLOBAL    MACRO     NAME,INDEX  TO INITIALIZE GLOBAL
          SX7       NAME
          SA7       B5+INDEX
          ENDM
          SPACE     5
SAVE      MACRO               PUT AT EACH ENTRY
          SA7       B6
          SX7       B6
          SA7       =XGLOBAL0
          ENDM
          SPACE     5
RETURN    MACRO
          RESET
          SA2       B6
          SB4       X2
          JP        B4
          ENDM
          TITLE                    COMPASS ROUTINES
*
*         REGISTER WHOMPING SUBROUTINES
*
          SPACE     5
          VFD       4/0,7/70B,7/52B,7/140B,35/0,60/0,60/0
XJ        SAVE
          SB4       B6+2        POINT AT CALL NAME
+         XJ        -4
-         VFD       30/0
```

```
          BX1       X6          IN CASE ANY RETURN
          RETURN
          SPACE     5
          VFD       4/0,7/70B,7/52B,7/46B,7/140B,2B/0,60/0,60/0
XJF       SAVE
          SB4       B6+3
+         XJ        -4
-         VFD       30/1
+         SB4       1
-         JP        L1
+         SB4       B0
L1        BX1       X6
          RESET
          EQ        B4,B0,L2
          JP        B6+2        JUMP TO FRETURN
L2        SA2       B6          NORMAL RETURN
          SB4       X2
          JP        B4
          EJECT
          SPACE     5
          VFD       4/0,7/42B,7/45B,7/41B,7/44B,7/140B,21/0,60/0,60/0
BEAD      SAVE
          SA1       B6+3        DIR ENTRY OR STRING DES
          SB1       X1
          SA1       B6+4        CAPABILITY
          SB7       X1
          SA1       B6+5        NAME
          SA2       B6+6        USER NAME
          SA3       B6+2        ACTION
          SB6       X3
+         XJ        BEADLOC
-         VFD       30/0
          RETURN
BEADSC    EQU       1
BEADLOC   VFD       60/BEADSC
          VFD       4/0,7/42B,7/45B,7/41B,7/44B,7/140B,21/0,60/0,60/0
          TITLE                       MASK TABLE
*
*         MASK  TABLE
*
          SPACE     5
          ENTRY     MASKTAB,ENTRY,XJ,XJF,BEAD
ONES      SET       60
MASKTAB   BSS       0
          DUP       61
ZEROS     SET       60-ONES
          VFD       ZEROS/0,ONES/-0
ONES      SET       ONES-1
          ENDD
          TITLE                       ENTRY POINTS
*
          SPACE     5
*
*         CODE TO BE EXECUTED AT BEGINNING
*
          SPACE     5
```

```
ENTRY       SB5         GLOBAL0
            SX7         B5+GLBSIZE+1
            SA7         B5          HIDE INITIAL STACK P
            RESET
            SX7         =XXJRET
            GLOBAL      NEXTCAP,1
            GLOBAL      (NEXTCAP+FRECAPS),2
+           JP          =XSTART
            END                                                    *
//DECK BCPLGD
//  DECLARATIONS TO LINK WITH HEAD

GLOBAL [
            FREECAP     :   1  // INDEX OF FIRST FREE CAPABILITY
            NCAPS       :   2  // NUMBER OF FREE ONES
    ]

EXTERNAL [
            XJ          // XJ(OPERATION,PAR1,...,PARN)
            XJF         // XJF(F-RET-LABEL,OPERATION,PAR1,...)
            BEAD        // BEAD(B6,B1,B7,X1,X2)
    ]

MANIFEST [
            ALLOC = 0
            READ =3
            WRITE =4
            SENDE = 5
            GETE = 6
            CBLK = 14
            PROBE = 15
            DELBLK = 16
            SPRETURN = 17
            REDSHP = 18
            DSPCLX = 19
            FRETRN = 20
   ]

GLOBAL [ // TO LINK WITH FSP AND IO
            FLIST       :   5  // POINTER TO FREE LIST
            OUTPUT      :   6  // STANDARD OUTPUT STREAM
            C6TO7       :   7  // DISPLAY CODE ASCII CONVERSION VECTOR
            C7TO6       :   8  // ASCII DISPLAY CODE  CONVERSION VECTOR
            IOBASE      :   9  // OPEN STREAM LIST POINTER
            MONITOR     :  10
    ]

EXTERNAL [
            NEWVEC          // NEWVEC(SIZEWANTED)
            RETVEC          // RETVEC(V,SIZEOFV)
            PACKSTRING      // (VECTOR,STRING)
            UNPACKSTRING    // (STRING,VECTOR)
            INITIALIZEIO
            FINDINPUT       //   (↓FNAME,USERNAME↓)
            CREATEOUTPUT    //   (↓FNAME,USERNAME↓)
            READCH          // (STREAM,LV CH)
```

```
        WRITECH         // (STREAM,CH)
        WRITES          // OUTPUT>STREAM; WRITES(STRING)
        WRITEN          // OUTPUT>STREAM; WRITEN(NUMBER)
        WRITEO          // OUTPUT>STREAM; WRITEO(NUMBER)
        READN           // NUMBER READER N>READN(STREAM)
        TRACE
        TRACECALL
        TRACERETURN
        DUMP
        ENDREAD         // CLOSE INPUT STREAM
        ENDWRITE        // CLOSE OUTPUT STREAM
        IOERROR         // GENERAL IOERROR ROUTINE
        UNPACKTO        //  (STR,VEC,BREAKCH)
        WRITEVEC
        READVEC
        FLUSH           // FORCE OUT A LINE WITHOUT CR
        BCDWORD
        ASCII
        ENDOFSTREAM
        CLOSEALL
        ABORT
    ]

MANIFEST [ AMASK = 777777B  ; ENDOFSTREAMCH =144B
           ESCAPECH = 173B ]
//END     *END
//DECK I01
GET "BCPLGD"
MANIFEST [
          TTYS =  0 // SIGNALS STREAM IS TTY
          LINK = 1  // TO CHAIN STREAMS TOGETHER
          INOROUT = 2 // INPUT OR OUTPUT
      ]
MANIFEST [ CHPTR= 3 ]
MANIFEST [ CBUFF = 4 ]
MANIFEST [ CBUFFSIZE=160 ]
MANIFEST [ PNAME = CBUFF+CBUFFSIZE ]
MANIFEST [ UNAME = PNAME+1 ]
MANIFEST [ CINDEX = UNAME+3 ]  // LEAVE 4-WORD BLOCK FOR READ
MANIFEST [ BBASE = CINDEX+1 ]
MANIFEST [ BSIZE = BBASE+1 ]
MANIFEST [ SIZE = BSIZE+1 ]
MANIFEST [ NWORD=SIZE+1 ]
MANIFEST [ NFA=NWORD+1 ]
MANIFEST [ NAME = NFA+1 ]

LET READNAME(S,PN,UN) BE
    [ LET V = VEC 20
      UNPACKSTRING(S,V)
      [ LET S=54 AND C=0 AND I=1
        UNTIL I>V.0 \ V.I=#,# DO
        [ IF S GE 0 DO C ! C \ C7TO6.(V,I) ^ S
          S,I ! S-6,I+1 ]
        RV(PN) ! C
        C,S,I ! 0,54, I LE V.0 -> I+1,I
        UNTIL I>V.0 DO
```

```
        [ IF S GE 0 DO C ← C ≤ C7TO6,(V,I) ↑ S
          I,S ← I+1,S-6 ]
        RV UN ← C  ]1

LET INITIALIZEIO(V, SIZE) BE
 [1 IOBASE ← 0; FLIST ← 0; RETVEC(V,SIZE+1)
    [ LET T=NEWVEC(NCAPS+1)
      FOR I=0 TO NCAPS DO T,I ← I GE FREECAP
      FREECAP ← T ]
   C7TO6 ← TABLE
55B,66B,71B,60B,53B,74B,67B,64B,51B,52B,47B,45B,56B,46B,57B,50B,
//     ≤   ↓   ≡   $   %   ^   ≠   (   )   *   +   ,   -   .   /
33B,34B,35B,36B,37B,40B,41B,42B,43B,44B,63B,77B,72B,54B,73B,75B,
//0   1   2   3   4   5   6   7   8   9   :   ;   <   =   >   ≥
00B,01B,02B,03B,04B,05B,06B,07B,10B,11B,12B,13B,14B,15B,16B,17B,
//     A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
20B,21B,22B,23B,24B,25B,26B,27B,30B,31B,32B,61B,76B,62B,70B,65B,
//P   Q   R   S   T   U   V   W   X   Y   Z   [   ∨   ]   ↑   ←
64B,01B,02B,03B,04B,05B,06B,07B,10B,11B,12B,13B,14B,15B,16B,17B,
//≠   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
20B,21B,22B,23B,24B,25B,26B,27B,30B,31B,32B,51B,00B,52B,55B,55B
//P   Q   R   S   T   U   V   W   X   Y   Z   (       )

    C6TO7 ← TABLE 0,
    ≠A≠,≠B≠,≠C≠,≠D≠,≠E≠,≠F≠,≠G≠,≠H≠,≠I≠,≠J≠,≠K≠,≠L≠,≠M≠,≠N≠,≠O≠,≠P≠,
    ≠Q≠,≠R≠,≠S≠,≠T≠,≠U≠,≠V≠,≠W≠,≠X≠,≠Y≠,≠Z≠,≠0≠,≠1≠,≠2≠,≠3≠,≠4≠,≠5≠,
    ≠6≠,≠7≠,≠8≠,≠9≠,≠+≠,≠-≠,≠*≠,≠/≠,≠(≠,≠)≠,≠$≠,≠=≠,≠ ≠,≠,≠,≠.≠,≠≡≠,
    ≠[≠,≠]≠,≠:≠,≠≠≠,≠←≠,≠≤≠,≠^≠,≠↑≠,≠∨≠,≠<≠,≠>≠,≠%≠,≠≥≠,≠∨≠,≠;≠
          ]1

LET NEWCAP() =
    VALOF [ FOR I=0 TO NCAPS
            DO [ IF FREECAP,I THEN [ FREECAP,I←FALSE; RESULTIS I]]
            IOERROR(3) ]

LET RETCAP(I) BE [ FREECAP,I ← TRUE ]

LET CREATEFDB(LFN) =
    VALOF [ LET V=VEC 50
            UNPACKSTRING(LFN,V)
          [ LET FDBS= NAME+V,0/8+1  //SIZE OF FDB
            LET ST = NEWVEC(FDBS)
            PACKSTRING(V,ST+NAME)
            BEADNAME(LFN,ST+PNAME,ST+UNAME)
            CINDEX,ST ← NEWCAP()
            BEAD(0,ST+PNAME,CINDEX,ST,PNAME,ST,UNAME,ST)
            XJ(REDSHP,CINDEX,ST,V,50)
            BSIZE,ST ← V,(V,0)
            BBASE,ST ← NEWVEC(BSIZE,ST)
            TTYS,ST,NWORD,ST,SIZE,ST←FALSE,BBASE,ST,FDBS
            RESULTIS ST  ]]

LET FINDINPUT(LFN) =
    VALOF [ LET ST=0  // TO BECOME STREAM-POINTER
            TEST LFN =0   // MEANS A TTY STREAM
              DO [ ST ← NEWVEC(CBUFF+CBUFFSIZE)
```

```
                TTYS.ST ↛ TRUE ]
          OR [ ST ↛ CREATEFDB(LFN)
               IF XJ(PROBE,CINDEX.ST,0)=0 THEN IOERROR(4)
               XJ(READ,CINDEX.ST,0,BBASE.ST,BSIZE.ST)
               NFA.ST ↛ BSIZE.ST ]
        INOROUT.ST,LINK.ST,CHPTR.ST ↛
        TRUE,       IOBASE,   0
        IOBASE ↛ ST
        RESULTIS ST  ]

LET READVEC(ST,V,COUNT,EL,EC) BE
  [RV LET N=NWORD.ST AND E=BBASE.ST+BSIZE.ST
      FOR I=0 TO COUNT-1 DO
      [ IF N GE E
          DO [ XJF(EOFL,READ,CINDEX.ST,NFA.ST,BBASE.ST,BSIZE.ST)
               NFA.ST ↛ NFA.ST +. BSIZE.ST; N↛BBASE.ST; GOTO OK
          EOFL: RV EC ↛I; GOTO EL
           OK :  ]
        V.I := RV N
        N ↛ N + 1 ]
      NWORD.ST ↛ N  ]RV

 LET READCH(ST,CH) BE
     [R LET CHP = CHPTR.ST
        IF CHP=0 THEN  // GET A NEW LINE
          [ LET V=VEC CBUFFSIZE/8
            TEST TTYS.ST
              DO [ BEAD(6,V)  // GET A LINE FROM TTY
                   UNPACKTO(V,CHPTR+ST,#*N#)
                   ST.(CBUFF+CHPTR.ST) ↛ #*N# ]
              OR [[ LET W=NWORD.ST
                    AND FIN=BBASE.ST+BSIZE.ST
                    AND I=0
                    WHILE RV W RSHIFT 56 = 14B DO // SKIP OVER SLOP
                     [S W↛W+(RV W ∧ AMASK)
                        IF W GE FIN THEN // IF SLOPPY BLOCK GOES
                                         // ACROSS A BLOCK BOUNDARY
                                         // WE HAVE TO FOOL AROUND
                         [ NFA.ST↛NFA.ST+.((W-FIN)/BSIZE.ST)*BSIZE.ST
                           W↛BBASE.ST+(W-FIN)MOD BBASE.ST
                           XJF(EOFL,READ,CINDEX.ST,NFA.ST,BBASE.ST,
                                                   BSIZE.ST)
                           NFA.ST↛NFA.ST+.BSIZE.ST ]S
                    IF RV W RSHIFT 56 = 10B THEN
                      [ RV CH ↛ ENDOFSTREAMCH; RETURN ]
                    [ V.I ↛ RV W
                      I,W ↛ I+1,W+1
                      IF I GE CBUFFSIZE/8 THEN IOERROR(3)
                      IF W GE FIN  // RE-FILL BUFFER
                        DO [ XJF(EOFL,READ,CINDEX.ST,NFA.ST,BBASE.ST,BSIZE.ST)
                             NFA.ST,W ↛ NFA.ST+.BSIZE.ST, BBASE.ST ]]
                    REPEATUNTIL RV W < 0  // NEGATIVE WORD BEGINS NEXT LINE
                    V.I ↛ #*N#↑49  // FOR GOOD LUCK
                    NWORD.ST ↛ W ]
                  [ LET J,S,C = 0,-49,0
                    [ C↛ RV V ↑ S ∧ 177B
```

```
                    TEST C=ESCAPECH
                      DO [ S:=S+7; IF S>0 THEN V!S := V+1!-49
                           C := RV V ! S & 177B
                           FOR Z=1 TO C DO
                            [ ST!(J+CBUFF) := # #; J:=J+1 ]
                           J := J-1 ]
                      OR ST!(J+CBUFF) := C
                    IF C=##N# BREAK
                    S!J := S+7!J+1
                    IF S > 0 DO V!S := V+1!-49
                    IF J GE CBUFFSIZE THEN IOERROR(3) ]
                REPEAT ]]
            CHP := ST+CBUFF ]
        RV CH := RV CHP
        CHPTR!ST := RV CH=##N# -> 0,CHP+1
        RETURN
EOFL: IOERROR(5) ]R

LET ENDREAD(ST) BE
    [ LET T=LV IOBASE
      UNTIL RV T=ST DO
        [ IF RV T = 0 THEN IOERROR(7)
          T := RV T + LINK ]
      RV T := LINK!(RV T)  // TAKE OUT OF ACTIVE STREAM LIST
      TEST TTYS!ST
        DO [ RETVEC(ST,CBUFF+CBUFFSIZE) ]
        OR [ RETVEC(BBASE!ST, BSIZE!ST)
             BEAD(1,PNAME+ST,CINDEX!ST)   // CLEAR BUSY BIT
             RETCAP(CINDEX!ST)
             RETVEC(ST,SIZE!ST) ]]

LET ENDOFSTREAM(ST) = RV(NWORD!ST) RSHIFT 56 = 10B

LET CREATEOUTPUT(LFN)=
    VALOF [ LET ST=0
            TEST LFN=0
              DO [ ST := NEWVEC(CBUFF+CBUFFSIZE)
                   TTYS!ST := TRUE ]
              OR [ ST := CREATEFDB(LFN)
                   NFA!ST := 0; ST!CBUFF := #NOTSYSTX# ] // SEE ENDWRITE
            CHPTR!ST,INOROUT!ST,LINK!ST :=
            0,        FALSE,       IOBASE
            IOBASE := ST
            RESULTIS ST ]

LET WRITEVEC(ST,V,COUNT) BE
    [1 LET N=NWORD!ST  // NEXT FREE WORD
      AND E=BBASE!ST+BSIZE!ST  // END OF BUFFER
      FOR I=0 TO COUNT-1 DO
       [2 RV N := V!I
          N := N+1
          IF N GE E
            DO [ L: XJF(MISSB,WRITE,CINDEX!ST,NFA!ST,BBASE!ST,BSIZE!ST)
                 NFA!ST := NFA!ST + BSIZE!ST
                 N := BBASE!ST
                 GOTO OK
```

```
            MISSB!XJ(CBLK,CINDEX,ST,NFA,ST)! GOTO L
              OK!  ]2
      NWORD.ST ! N ]1

LET FLUSH(ST) BE
 [FLS UNLESS CHPTR.ST=0 DO
    [ LET V=VEC CBUFFSIZE
      AND S,I,BCNT,CP = 49,0,0,0
      V.0 ! 1B↑56
      FOR J=0 TO CHPTR.ST-1 DO
        [ LET C=ST.(CBUFF+J)
          TEST C=≠ ≠ THEN BCNT!BCNT+1 OR
          TEST C=≠*T≠ THEN [ LET SK=10-CP MOD 10
                             BCNT ! BCNT+SK
                             CP ! CP+SK-1 ] OR
          [ UNTIL BCNT=0
              DO [ TEST BCNT=1
                     DO [ V.I ! V.I ≤ ≠ ≠↑S! S!S-7
                          BCNT ! 0
                          IF S<0 DO I,S,V.I ! I+1,49,0 ]
                     OR [ V.I ! V.I ≤ ESCAPECH↑S! S!S-7
                          IF S<0 DO I,S,V.I ! I+1,49,0
                          [ LET T= BCNT>127 -> 127, BCNT
                            V.I ! V.I ≤ T↑S!S!S-7
                            BCNT ! BCNT - T ]
                          IF S<0 DO I,S,V.I ! I+1,49,0 ]
                   ]
          V.I ! V.I ≤ C↑S
          S ! S-7
          IF S<0 THEN I,S,V.I ! I+1,49,0 ]
        CP ! CP+1 ]
      // NOW THE LINE IS ALL PACKED UP AND READY
      TEST TTYS.ST  // WHERE IS IT GOING
        DO [ LET X=VEC 2
             X.0,X.1,X.2 ! V.I*8+(49-S)/7,CBUFFSIZE
             BEAD(5,X)  ]
        OR WRITEVEC(ST,V,I-S/49+1)
      CHPTR.ST ! 0 ]FLS

LET WRITECH(ST,CH) BE
 [WCH LET CHP = CHPTR.ST
      IF CHP GE CBUFFSIZE THEN IOERROR(6)
      ST.(CBUFF+CHP) ! CH
      CHPTR.ST ! CHP+1
      IF CH=≠*N≠ THEN FLUSH(ST)  ]WCH

LET ENDWRITE(ST) BE
    [ LET T=LV IOBASE
      UNTIL RV T=ST DO [ IF RV T = 0 DO IOERROR(7)
                         T ! RV T + LINK ]
      RV T ! LINK.(RV T)
      FLUSH(ST)
      TEST TTYS.ST
        DO RETVEC(ST,CBUFF+CBUFFSIZE)
        OR [ LET X = 10B↑56
             UNLESS ST.CBUFF=.≠NOTSYSTX≠ // FIRST WRITECH DESTROYS
```

```
                DO [ WRITEVEC(ST, LV X, I)
                     PNAME.ST ! PNAME.ST ≤ 200B ] // NOT SYSTEXT
              UNLESS NWORD.ST=BBASE.ST
               DO [ L: XJF(MISSB,WRITE,CINDEX.ST,NFA.ST,BBASE.ST,BSIZE.ST)
                       NFA.ST ! NFA.ST +.BSIZE.ST; GOTO OK
                MISSB: XJ(CBLK,CINDEX.ST,NFA.ST); GOTO L
                   OK: ]
              [ LET I=NFA.ST
                UNTIL XJ(PROBE,CINDEX.ST,I)=0
                 DO [ XJ(DELBLK,CINDEX.ST,I)
                      I ! I+.BSIZE.ST ]]
              RETVEC(BBASE.ST,BSIZE.ST)
              BEAD(1,PNAME+ST,CINDEX.ST)
              RETCAP(CINDEX.ST)
              RETVEC(ST,SIZE.ST) ]]
//END     *END
//DECK IO2
GET ↓BCPLGD↓

MANIFEST [ LINK=1; INOROUT=2 ]

LET CLOSEALL() BE
    [ UNTIL IOBASE = 0
     DO     (INOROUT.IOBASE => ENDREAD,ENDWRITE)(IOBASE) ]

LET ABORT() BE
     [ CLOSEALL() ; XJ(FRETRN)  ]

LET IOERROR(N) BE
    [ LET V=VEC 2
      V.1,V.2 ! 100,100
      V.0 ! ↓I/O ERROR.*N↓
      BEAD(5,V)
      V.0 ! N=0->↓BAD RETURN TO FREE SPACE.*N↓,
            N=1->↓FREE SPACE EXHAUSTED.*N↓,
            N=2->↓SYSTEXT LINE TO LONG.*N↓,
            N=3->↓FREE CAPABILITIES EXHAUSTED.*N↓,
            N=4->↓EMPTY INPUT FILE.*N↓,
            N=5->↓UNEXPEXTED END-OF-FILE*N↓,
            N=6->↓WRITING TOO LONG SYSTEXT LINE.*N↓,
            N=7->↓ATTEMPT TO CLOSE NON-STREAM.*N↓,
            N=8->↓UNPACKSTRING RUNNING WILD.*N↓,
             ↓UNKNOWN I/O ERROR.*N↓
        BEAD(5,V)
      ABORT() ]


LET NEWVEC(M)=
    VALOF[1 IF FLIST=0 THEN IOERROR(1)
            FLIST ! RV FLIST ^ AMASK
         [ LET OFL=FLIST
           [ LET P= RV FLIST ^ AMASK
             LET S= RV P ↑ -18
             IF S=M
              DO[ LET N= RV P ∧ AMASK
                  TEST N=P DO FLIST ! 0
```

```
                              OR RV FLIST ↛ RV FLIST ∧ =AMASK≤N
                    RESULTIS P  ]
               IF S >M
                DO[ RV P ↛ RV P =. (M↑18)
                    RESULTIS P+(S=M) ]
               FLIST ↛ P
               IF FLIST=OFL THEN IOERROR(1) ]
          REPEAT  ]1

LET RETVEC(V,M) BE
   [1 LET A=FLIST
      V ↛ V ∧ AMASK
      IF A=0 [ FLIST↛V; RV V ↛ M↑18≤V; RETURN ]
  L; [ LET MA = RV A ↑ -18
       AND B = RV A ∧ AMASK
       LET AL = A+MA
       AND VL=V+M
       IF AL<B => AL LE V ∧ VL LE B, AL LE V ≤ VL LE B
       DO [ TEST AL=V
              DO[ RV A ↛ RV A +. (M↑18); V↛A ]
              OR[ RV A↛ RV A ∧ NOT AMASK ≤ V
                  RV V ↛ M↑18 ≤ B ]
            IF VL=B
             DO[ RV V ↛ RV B +. (RV V ∧ NOT AMASK)
                 FLIST ↛ V ] // IN CASE IT WAS B
            RETURN ]
     A↛B; UNLESS A=FLIST GOTO L; IOERROR(0)  ]1

LET PACKSTRING(V,S) BE
    [1 LET I=49
       RV S ↛ 0
       FOR J=1 TO V.0 DO
         [ RV S ↛ RV S ≤ V.J ↑ I
           I ↛ I-7
           IF I<0 DO
           [ S ↛ S+1; RV S ↛ 0; I ↛ 49 ]]
       RV S ↛ RV S ≤ 140B↑I ]1

LET UNPACKTO(S,V,CH) BE
    [ LET I=0 AND J=-49 AND C=0
      [ C↛ RV S ↑ J ∧ 177B
        IF C=CH BREAK
        I↛I+1
        V.I ↛ C
        J↛J+7
        IF V.I GE LV S - 2 DO IOERROR(8)
        IF J>0 DO [ S↛S+1; J↛-49 ] ]
      REPEAT
      V.0 ↛ I  ]

LET UNPACKSTRING(S,V) BE
   [ UNPACKTO(S,V,140B) ]

 LET BCDWORD(S) = VALOF
        [ LET W = 0
          LET V = VEC 512
```

```
          UNPACKSTRING(S, V)
          [ LET I=54
            AND J=1
            UNTIL I<0 ≤ J>V.0 DO
              [ W := W ≤ C7TO6.(V.J) LSHIFT I
                I,J := I-6,J+1 ]  ]
           RESULTIS W  ]

LET ASCII(W,S) BE
    [ LET V= VEC 10 AND I=54 AND J=0
      [ LET C= W RSHIFT I ∧ 77B
        IF C=0 BREAK
        J := J+1
        V.J := C6TO7.C
        I := I-6  ]
      REPEATUNTIL I<0
      V.0 := J
      PACKSTRING(V,S)  ]

 LET WRITES(S) BE
        [ LET V = VEC 200
          UNPACKSTRING(S, V)
          FOR I = 1 TO V.(0) DO WRITECH(OUTPUT, V.(I))  ]

 LET WRITEN(N) BE
     [ IF N<0 THEN [ WRITECH(OUTPUT,≠-≠); N:=-N ]
       IF N=0 THEN [ WRITECH(OUTPUT,≠0≠); RETURN ]
     [ LET TENTAB = TABLE 1,10,100,1000,10000, 100000,1000000,
                    10000000,
                    100000000,
                    1000000000,
                    10000000000,
                    100000000000,
                    1000000000000,
                    10000000000000,
                    100000000000000,
                    1000000000000000
       AND I=1
       UNTIL TENTAB.I>N ≤ I>17 DO I:=I+1
       [ I:=I-1
         [ LET J=-1 AND X=TENTAB.I
           [ N:=N-X; J:=J+1 ] REPEATUNTIL N<0
           WRITECH(OUTPUT,J+≠0≠)
           N:=N+X ]]
       REPEATUNTIL I=0  ]]

 LET WRITEO(N) BE
        [ LET I = 60
          UNTIL I=0 DO  [ I := I-3
                          WRITECH(OUTPUT, ≠0≠ + (N↑-I ^ 7)) ] ]

LET READN(STREAM) =
   VALOF [ LET C,N1,N2,NEGA = 0,0,0,FALSE
           READCH(STREAM,LV C) REPEATUNTIL ≠0≠ LE C LE ≠9≠ ≤ C=≠-≠
```

```
        IF C=#-# DO [ NEGA:=TRUE; READCH(STREAM,LV C) ]
        [ N1 := 10*N1 +. (C-#0#)
          N2 := 8*N2 +. (C-#0#)
          READCH(STREAM,LV C) ]
        REPEATWHILE #0# LE C LE #9#
        IF NEGA THEN N1,N2 := -N1,-N2
        RESULTIS C=#B# -> N2,N1 ]
//END     *END
//DECK MANFEST
MANIFEST  // A E OPERATORS, OCODE OPERATORS, AND CANONICAL SYMBOLS

      [  ASS=50;
         CHARCONST=19;        COLON=54; COMMA=38; COND=37;  CONSTDEF=43;
         DATALAB=100;         DIV=124;
         QEND=90;  ENTRY=94;
         FNAP=10;  FNDEF=44; FNRN=96;
         ITEML=101;           ITEMN=102;
         JF=87;    JT=86;     JUMP=85;
         LAB=90;   LABEL=78; LG=41;    LL=44;    LLG=46;   LLL=47;
         LLP=45;   LMINUS=26;          LN=42;    LOCAL=77; LP=40;
         LPLUS=18; LSTR=43;  LX=201;
         MINUS=15; MULT=11;
         NAME=2;   NEG=17;   NUMBER=1;
         OCTAL=109;
         PLUS=14;  POS=16;
         QAND=40;  QASHIFT=65;         QBE=89;   QBREAK=66;
         QCASE=71; QDEFAULT=72;        QDO=101;  QEQ=20;   QEQV=35;
         QFALSE=5; QFINISH=68;         QFOR=86;  QGE=25;   QGET=87;
         QGLOBAL=76;          QGOTO=52; QGR=23;   QIF=57;   QINTO=98;
         QLE=24;   QLET=74;  QLEQ=27;  QLGE=49;  QLGR=29;  QLLE=64;
         QLLS=48;  QLNE=28;  QLOGAND=33;       QLOGOR=34;
         QLS=22;   QLSHIFT=31;         QLV=7;    QMANIFEST=75;
         QNE=21;   QNEQV=36; QNOT=30;  QOR=102;  QREM=13;  QREPEAT=61;
         QREPEATUNTIL=63;     QREPEATWHILE=62;   QRESULTIS=53;
         QRETURN=67;          QRSHIFT=32;        QRV=8;    QSTATIC=79;
         QSWITCHON=70;        QTABLE=39;         QTEST=55; QTHEN=100;
         QTO=99;   QTRUE=4;  QUNLESS=58;        QUNTIL=60;
         QVALOF=6; QVEC=103; QWHILE=59;
         RBRA=105; REL=19;   RES=98;   RESLAB=99;          RKET=106;
         RSTACK=93;           RTAP=51;  RTDEF=45; RTRN=97;
         SAVE=95;  SECTBRA=91;         SECTKET=92;         SEMICOLON=97
         SEQ=73;   SG=81;    SL=82;    SP=80;    STACK=91; STIND=83;
         STORE=92; STRINGCONST=3;
         VALDEF=41;           VECAP=9;  VECDEF=42;
         WORD=88;
         XTERNAL=202 ]

 MANIFEST  // CONSTANTS USED BY NEXTSYMB
   [ EMPTY=0; SIMPLE=1; IGNORABLE=2
     DIGIT=4; CAPITAL=5; SMALL=6; UNDERLINE=7  ]

MANIFEST   // CONSTANTS FOR REPRESENTATION OF STRINGS
   [ BYTEBITS=7; BYTEMAX=127; BYTESPERWORD=8  ]

MANIFEST [ AMASK=777777B ]
```

```
//END       *END
//DECK MISCGD
GLOBAL      //  MISC HEAD IS REFERENCED BY: CAE0, CAE2, I02, PLIST, TRN0,
      //                                          INPP3 AND TRN7.
       [  MESSAGE          : 11   // POINTS TO VECTOR OF ERROR MESSAGES
          REPORTCOUNT      : 12   // TOTAL NUMBER OF SYNTACTIC ERRORS
          PARAMNUMBER      : 13   //
          CREATED          : 14   //
]     //        GLOBAL FOUND DIRECTLY IN CODE

EXTERNAL [
          COMPILEAE                // MAIN TRANS ROUTINE
          OVERLAY                  // USED TO LINK CODE GENERATOR
          PLIST                    // PRINTS A E TREE
         ]
//END       *END
//DECK CAEGD
GLOBAL      // CAEGD       ( ALL INPP AND CAE FILES USE THIS HEAD )
      //
       [  B                : 15   // SIGNALS INSERTION OF NSYMB BY NEXTSYM
          CH               : 16   // 1 CHARACTER INPUT BUFFER
          CHKIND           : 17   // KIND OF CHARACTER IN CH : DIGIT, ETC.
          ERRORNAME        : 18   // SUBSTITUTED FOR UNKNOWN IDENTIFIER
          FREELISTP        : 19   // FREE-STORAGE FOR THE A E TREE
          GETP             : 20   // STACK POINTER FOR STACK OF INPUT FILE
          GETV             : 21   // STACK OF INPUT FILES (USED FOR GET)
          INPUT            : 22   // POINTER TO INPUT STREAM
          LINEP            : 23   // CHARACTER POSITION IN INPUT LINE
          LINEV            : 24   // CURRENT INPUT LINE
          NAMETREE         : 25   // TREE OF IDENTIFIERS
          NLPENDING        : 26   // TRUE IFF MORE INPUT
          NSYMB            : 27   // FOR INSERTION OF TOKENS (≠1≠,QDO,)
          REPORT           : 28   // SIGNALS SYNTAX ERRORS
          ST               : 29   //
          SYMB             : 30   // CURRENT TOKEN TYPE
          SYNERROR         : 31   // TYPE AND POSITION OF SYNTAX ERROR
          V                : 32   // CONTAINS CURRENT TOKEN
          VP               : 33   // LENGTH OF TOKEN IN V
          WORDV            : 34 ] //

EXTERNAL [
          FORMNUMBER               // CHANGES CHARACTER TO NUMBER (-≠0≠)
          KIND                     // DETERMINES KIND OF CH (DIGIT,LETTER,)
          LIST1                    // CREATES 1 WORD NODE ON A E TREE
          LIST2                    // CREATES 2 WORD NODE ON A E TREE
          LIST3                    // CREATES 3 WORD NODE ON A E TREE
          LIST4                    // CREATES 4 WORD NODE ON A E TREE
          LIST5                    // CREATES 5 WORD NODE ON A E TREE
          LOOKUPWORD               // CHECKS FOR RESERVED WORDS
          NEXTSYMB                 // PARSES NEXT TOKEN
          PP0                      // USED BY NEXTSYMB
          PP1                      // USED BY NEXTSYMB
          PP2                      // USED BY NEXTSYMB
          RBLOCK                   // TOP DOWN BLOCK RECOGNIZER
          RCH                      // PUT NEXT CHARACTER IN CH
          RCOM                     // TOP DOWN COMMAND RECOGNIZER
```

```
          RDEF                  // TOP DOWN DEFINITION RECOGNIZER
          READBLOCKBODY         // THE NAME SPEAKS FOR ITSELF
          REXP                  // TOP DOWN EXPRESSION RECOGNIZER
          RNAME                 // TOP DOWN NAME RECOGNIZER
          RNAMELIST             // TOP DOWN NAMELIST RECOGNIZER
         ]

MANIFEST[
          VMAX = 128              // MAXIMUM SIZE FOR A TOKEN (CHARS)
                                 ]

//END       *END
//DECK CAE0
GET *BCPLGD*
GET *CAEGD*
GET *MISCGD*
GET *MANFEST*
LET START() BE
[1  LET CLOCK = VEC 4
    AND FREECAPSAVE = FREECAP
    AND VERSION =

              *BCPL:2*

 RESTART: XJ(DSPCLX+CLOCK)
          FREECAP := FREECAPSAVE

        [ LET INAME = 0
          AND ONAME = *OCODE*
          AND TIME = FALSE
          AND ABORTFLG = TRUE
          AND BNAME = 0
          AND CNAME = 0
          AND CHREENT = FALSE
          AND COMPASS = FALSE
          AND BINARY = TRUE
          AND IDENT = 0

[A        LET VV = VEC 6000
          INITIALIZEIO(VV,6000)
          MONITOR := CREATEOUTPUT(0)
          OUTPUT := MONITOR

          WRITES(VERSION); WRITES(* LISTENING*N*)

//        INPUT PARSER FOR CONTROL CARD
//
          INPUT := FINDINPUT(0)
          [2        LET PARAM(CHAR) = VALOF
                    [         LET NAME = VEC 20
                              AND P = NEWVEC(2)
                              AND COUNT = 0
                    [         READCH(INPUT,CHAR)
                              IF CHAR.0=#;# <= CHAR.0=#*N# DO
                              [         NAME.0 := COUNT
                                        PACKSTRING(NAME,P)
```

```
                        RESULTIS P ]
                UNLESS CHAR.0=≠ ≠ DO
                        COUNT, NAME.COUNT ↱ COUNT + 1, CHAR.0 ]
                                REPEAT ]
        AND CHAR = 0 AND CHAR2 = 0
THETOP: READCH(INPUT,LV CHAR); READCH(INPUT, LV CHAR2)
  TOP:  TEST CHAR = ≠I≠ THEN
                INAME ↱ PARAM(LV CHAR2) OR
        TEST CHAR = ≠B≠ THEN
                BNAME ↱ PARAM(LV CHAR2) OR
        TEST CHAR = ≠N≠ THEN
                IDENT ↱ PARAM(LV CHAR2) OR
        TEST CHAR = ≠C≠ ∧ CHAR2 = ≠=≠ THEN
                CNAME ↱ PARAM(LV CHAR2) OR
        TEST CHAR = ≠T≠ THEN
                TIME ↱ TRUE OR
        TEST CHAR = ≠O≠ THEN
                ONAME ↱ PARAM(LV CHAR2) OR
        TEST CHAR = ≠S≠ THEN
        [       TEST CHAR2 = ≠A≠ THEN
                        ABORTFLG ↱ FALSE OR
                        ABORT()
                READCH(INPUT,LV CHAR2) ] OR
        TEST CHAR = ≠C≠ THEN
        [       TEST CHAR2 = ≠R≠ THEN
                        CHREENT ↱ TRUE OR
                        ABORT()
                READCH(INPUT,LV CHAR2) ] OR
        TEST CHAR = ≠*N≠ ≤ CHAR = ≠ ≠ THEN
        [       CHAR ↱ CHAR2
                READCH(INPUT,LV CHAR2)
                GOTO TOP ] OR ABORT()
        IF CHAR2=≠;≠ GOTO THETOP
        UNLESS CNAME=0 DO COMPASS ↱ TRUE
        IF CNAME.0=↓TTY↓.0 DO CNAME.0 ↱ 0
        IF INAME.0=↓TTY↓.0 DO INAME ↱ 0
        IF BNAME.0=↓0↓.0 DO BINARY ↱ FALSE
        IF CNAME.0=↓0↓.0 DO COMPASS ↱ FALSE
        IF IDENT=0 DO IDENT ↱ (INAME=0)->↓BCPL↓,INAME
        IF BNAME=0 DO
        [       LET V = VEC 20
                UNPACKSTRING((INAME=0->↓BCPL↓,INAME),V)
                FOR I = 0 TO 19 DO
                        V.(20-I) ↱ V.(19-I)
                V.1, V.0 ↱ ≠B≠, V.0+1
                BNAME ↱ NEWVEC(2)
                PACKSTRING(V,BNAME) ]2
    ENDREAD(INPUT)
L:  [ LET A = 0
    AND FORMNUMBER(X) = X-≠0≠       // X IS AN ASCII DIGIT

    AND LIST1(X) = VALOF
    [       LET V = NEWVEC(1)
            FREELISTP ↱ FREELISTP + 1
            V.(0) ↱ X
            RESULTIS V  ]
```

```
    AND LIST2(X,Y) = VALOF
    [           LET V = NEWVEC(2)
                FREELISTP ! FREELISTP + 2
                V!(0), V!(1) ! X, Y
                RESULTIS V  ]

    AND LIST3(X,Y,Z) = VALOF
    [           LET V = NEWVEC(1)
                FREELISTP ! FREELISTP + 1
                V!(0) ! (X^AMASK)<(Y^AMASK)<<18<Z LSHIFT 36
                RESULTIS V  ]

    AND LIST4(X,Y,Z,T) = VALOF
    [           LET V = NEWVEC(2)
                FREELISTP ! FREELISTP + 2
                V!0 ! (X^AMASK) < (Y^AMASK)<<18 < Z LSHIFT 36
                V!1 ! T
                RESULTIS V  ]

    AND LIST5(X,Y,Z,T,U) = VALOF
    [           LET V = NEWVEC(2)
                FREELISTP ! FREELISTP + 2
                V!0 ! (X^AMASK) < (Y^AMASK)<<18 < Z LSHIFT 36
                V!1 ! (T^AMASK) < (U^AMASK)<<18
                RESULTIS V  ]
    FREELISTP ! 0

[2  LET R = REPORT
    LET CAEREPORT(N) BE
            [ REPORTCOUNT ! REPORTCOUNT + 1
              IF SYNERROR = 0 THEN SYNERROR ! N*100+LINEP ]

    LET V1 = VEC 90
    LINEV, LINEP ! V1, 0
[   LET V1 = VEC 2
    GETV, GETP ! V1, 0
[   LET V1 = VEC 50
    WORDV ! V1
    MESSAGE ! TABLE
         *ILLEGAL SYMBOL; NAME EXPECTED.*,
         *ERROR AFTER MANIFEST OR GLOBAL*,
         *= OR: EXPECTED IN MANIFEST OR GLOBAL DEFINITION*,
         *SECTION BRACKET MISSING AT END OF A MANIFEST OR GLOBAL *,
         *ERROR AT START OF BLOCK; SECTION BRACKET MISSING.*,
         *SECTION BRACKET MISSING AT END OF BLOCK.*,
         *NAME MISSING IN A NAME LIST.*,
         *NAME TOO LONG (GREATER THAN 19 CHARACTERS LONG).*,
         *CLOSING BRACKET EXPECTED.*,
         *BRACKET MISSING AT END OF ARGUMENT LIST.*,
         *CONDITIONAL EXPRESSION COMMA MISSING.*,
         *EXPRESSION MISSING.*,
         *ERROR IN THE HEADING OF A FUNCTION OR ROUTINE DEF.*,
         *BRACKET MISSING AT THE END OF A FORMAL PARAMETER LIST.*,
         *= OR BE EXPECTED IN DEFINITION.*,
         *ERROR IN VECTOR DEFINITION.*,
```

```
            ↓DEFINING OPERATOR MISSING; THE OPERATOR = IS ASSUMED.↓,
            ↓; OUT OF CONTEXT IN COMMAND.↓,
            ↓ERROR IN COMMAND; POSSIBLY ↑ MISSING. ↓,
            ↓DO (OR THEN) MISSING IN AN; IF, UNLESS, WHILE OR UNTIL↓,
            ↓THEN (OR DO) MISSING IN TEST COMMAND.↓,
            ↓OR MISSING IN TEST COMMAND.↓,
            ↓= MISSING IN FOR COMMAND.↓,
            ↓TO MISSING IN FOR COMMAND↓,
            ↓DO (OR THEN) MISSING IN FOR.↓,
            ↓INTO MISSING IN A SWITCHON COMMAND.↓,
            ↓; MISSING IN A CASE LABEL↓,
            ↓; MISSING AFTER DEFAULT.↓,
            ↓NAME MISSING AFTER FOR.↓,
            ↓ERROR IN OCTAL CONSTANT↓,
            ↓CHARACTER CONSTANT TOO LONG↓,
            ↓ILLEGAL CHARACTER, THE CHARACTER IS IGNORED.↓,
            ↓STRING CONSTANT TOO LONG.↓,
            ↓SYNTAX ANALYSIS COMPLETE BEFORE END OF SOURCE TEXT↓,
            ↓STRING EXPECTED AFTER GET↓
//
//  NEXT ERROR MESSAGE = 35
//
        REPORTCOUNT, SYNERROR ↑ 0,0
        INPUT ↑ FINDINPUT(INAME)
        REPORT, NAMETREE ↑ CAEREPORT, 0
        ERRORNAME ↑ TABLE NAME, 0, 0, ≠$S*0000     ≠

    [ LET V1 = VEC VMAX
      V,VP,SYMB,NSYMB,ST,NLPENDING ↑ V1,0,0,0,0,TRUE
      CH, CHKIND, B ↑ 0, EMPTY, FALSE
M:  NEXTSYMB()

P:  A ↑ READBLOCKBODY()
      UNLESS SYMB=QEND DO
              [ IF REPORTCOUNT = 0 DO REPORT(3)
                NEXTSYMB()
                GOTO P    ]
      IF SYNERROR ≡ 0 THEN
                [ LET OUT = OUTPUT
                          OUTPUT ↑ MONITOR
                          WRITES(MESSAGE . (SYNERROR/100))
                          OUTPUT ↑ OUT ]
      ENDREAD(INPUT)
      REPORT ↑ R  ]

      WRITES( ↓*TAE TREE SIZE = ↓ ); WRITEN(FREELISTP)
      WRITECH(OUTPUT,≠*N≠) ]2

      IF REPORTCOUNT > 0 THEN
                IF ABORTFLG GOTO RESTART

      OUTPUT ↑ CREATEOUTPUT(ONAME)
      COMPILEAE(A)
      ENDWRITE(OUTPUT)
      OUTPUT ↑ MONITOR
      IF REPORTCOUNT NE 0 ∧ ABORTFLG GOTO RESTART
```

```
          CLOSEALL() ]A
           IF COMPASS ≤ BINARY
                      THEN OVERLAY(BINARY,COMPASS,CHREENT,BNAME,CNAME,
                                                  IDENT,ONAME)
          IF TIME DO
          [          LET C = VEC 4 AND VV = VEC 500
                     INITIALIZEIO(VV,500)
                     OUTPUT ← CREATEOUTPUT(0)
                     XJ(DSPCLX,C)
                     WRITECH(OUTPUT,≠*T≠)
                     FOR I = 1 TO 3 DO
                     [          WRITEN(C,I-CLOCK,I); WRITES(↓, ↓) ]
                     WRITECH(OUTPUT,≠*N≠); CLOSEALL() ]
          GOTO RESTART ]1
//END       *END
//DECK CAE1
GET ↓BCPLGD↓
GET ↓CAEGD↓
GET ↓MANFEST↓

 LET READBLOCKBODY() = VALOF
     [1 LET A, B, CDEFS, OP = 0, 0, 0, 0
        SWITCHON SYMB INTO
         [ CASE QMANIFEST:
           CASE QSTATIC:
           CASE QGLOBAL:
           CASE XTERNAL:
[2 LET N = 0
   LET V1 = VEC 20
   OP ← SYMB
   NEXTSYMB()
   UNLESS SYMB=SECTBRA DO
                        [ REPORT(1)
                          RESULTIS 0  ]
   N, V1.(0) ← VP, VP
   FOR I = 1 TO N DO V1.(I) ← V.(I)
   NEXTSYMB()

L: TEST SYMB=NAME THEN A ← RNAME()
                  OR  [ REPORT(0)
                        A ← ERRORNAME
                        NEXTSYMB()  ]
   TEST SYMB=(OP=QGLOBAL=>COLON;QEQ) THEN
      [3    NEXTSYMB()
            B ← REXP(0)
            IF OP=XTERNAL DO
                 TEST (B.0^AMASK)=STRINGCONST
                      THEN B ← B + 1
                      ELSE [ REPORT(1); B←A+3 ]3
   OR TEST OP=XTERNAL THEN B ← A + 3
   OR [ REPORT(2); NEXTSYMB(); B ← REXP(0) ]
   CDEFS ← LIST4(CONSTDEF, CDEFS, A, B)
   IF SYMB=SEMICOLON DO
                        [ NEXTSYMB()
                          GOTO L  ]
```

```
                UNLESS SYMB=SECTKET DO
                                    [ REPORT(3)
                                      RESULTIS 0  ]
                V.(0) := VP
                IF EQVEC(V1, V, N) DO NEXTSYMB()   ]2

                A := READBLOCKBODY()
                RESULTIS LIST3(OP, CDEFS, A)

        CASE QLET: NEXTSYMB()
                A := RDEF()
                B := READBLOCKBODY()
                RESULTIS LIST3(QLET, A, B)

        CASE SECTKET:
        CASE QEND:RESULTIS A

        CASE SEMICOLON: NEXTSYMB()
        DEFAULT:  A := RCOM()
                WHILE SYMB=SEMICOLON DO
                        [ NEXTSYMB()
                          B := RCOM()
                          A := LIST3(SEQ, A, B)  ]
                RESULTIS A   ]1

AND RBLOCK() = VALOF
        [ LET A, N = 0, 0
          LET V1 = VEC 20
          UNLESS SYMB=SECTBRA DO
                  [ REPORT(4)
                    RESULTIS 0  ]
          N, V1.(0) := VP, VP
          FOR I = 1 TO N DO V1.(I) := V.(I)
          NEXTSYMB()
          A := READBLOCKBODY()
          UNLESS SYMB=SECTKET DO
                  [ REPORT(5)
                    [ IF SYMB=QEND RESULTIS A
                      NEXTSYMB()  ]
                    REPEATUNTIL SYMB=SECTKET  ]
          V.(0) := VP
          IF EQVEC(V1, V, N) DO NEXTSYMB()
          RESULTIS A  ]

AND RNAMELIST() = VALOF
        [ LET A, B, C = 0, 0, 0
          UNLESS SYMB=NAME DO  [ REPORT(6)
                                 NEXTSYMB()
                                 RESULTIS ERRORNAME  ]
          A := RNAME()
          UNLESS SYMB=COMMA RESULTIS A
          NEXTSYMB()
          B := RNAMELIST()
          RESULTIS LIST3(COMMA, A, B)  ]

AND RNAME() = VALOF
```

```
   [1 LET A, N = 0, 0
      LET S = VEC 3
      IF VP GR 20 DO  [ REPORT(7)
                        VP ← 20  ]
      N, V.(0) ← VP, VP
      PACKSTRING(V, S)
      NEXTSYMB()
    [ LET M = LV NAMETREE
      LET X = S.(0)
      LET L = 0
      N ← N/BYTESPERWORD + 1


NEXT: L ← RV M
      IF L NE 0 DO
              [2 LET Y = L.(3)

                 TEST X GR Y THEN M ← L + 1
                                 OR M ← L + 2
                 UNLESS X=Y GOTO NEXT

                 FOR I = 1 TO N-1 DO UNLESS L.(I+3)=S.(I) GOTO NEXT

                 RESULTIS L   ]2

      L ← NEWVEC(N+3); FREELISTP ← FREELISTP + N + 3
      L.(0), L.(1), L.(2) ← NAME, 0, 0

      FOR I = 0 TO N-1 DO L.(I+3) ← S.(I)
      RV M ← L
      RESULTIS L   ]1

 AND EQVEC(V1, V2, N) = VALOF
                    [ FOR I = 0 TO N IF V1.(I) NE V2.(I) RESULTIS FALSE
                      RESULTIS TRUE  ]

//END       *END
//DECK CAE2
GET ↓BCPLGD↓
GET ↓CAEGD↓
GET ↓MISCGD↓
GET ↓MANFEST↓
 LET REXP(N) = VALOF
     [1 LET A, B, C, OP = 0, 0, 0, 0


        IF SYMB=NAME DO   // THIS IS THE USUAL CASE
                    [ A ← RNAME()
                      GOTO L  ]

        SWITCHON SYMB INTO
        [A DEFAULT:  REPORT(11)
                     NEXTSYMB()
                     RESULTIS LIST2(NUMBER, 0)

           CASE STRINGCONST:
```

```
CASE CHARCONST:
        [ LET S = VEC 20
          IF SYMB=CHARCONST DO
                  [ LET X=0
                    FOR I=1 TO VP DO
                    X := X * BYTEBITS + V.I
                    A := LIST2(NUMBER,X)
                      NEXTSYMB()
                      GOTO L  ]
          V.(0) := VP
          PACKSTRING(V, S)
        [ LET N = VP/BYTESPERWORD + 1
          A := NEWVEC(N+1); FREELISTP := FREELISTP + N + 1
          A.(0) := STRINGCONST
          FOR I = 0 TO N-1 DO A.(I+1) := S.(I)
          NEXTSYMB()
          GOTO L  ]  ]

CASE NUMBER: A := 0
             TEST RV V = OCTAL
               THEN FOR I=1 TO VP DO A := A * 3 LOGOR
                                          FORMNUMBER(V.(I))
          OR FOR I = 1 TO VP DO A := A*10 +  FORMNUMBER(V.(I))
             NEXTSYMB()
             A := LIST2(NUMBER, A)
             GOTO L

CASE QTRUE:
CASE QFALSE: A := LIST1(SYMB)
             NEXTSYMB()
             GOTO L

CASE RBRA:   NEXTSYMB()
             A := REXP(0)
             UNLESS SYMB=RKET DO REPORT(8)
             NEXTSYMB()
             GOTO L

CASE QVALOF: NEXTSYMB()
             A := RBLOCK()
             A := LIST3(QVALOF, A, 0)
             GOTO L

CASE QLV:
CASE QRV:   OP := SYMB
           NEXTSYMB()
           A := REXP(35)
           A := LIST3(OP, A, 0)
           GOTO L

CASE LPLUS:
CASE PLUS: NEXTSYMB()
           A := REXP(34)
           GOTO L

CASE LMINUS:
```

```
      CASE MINUS: NEXTSYMB()
                  A := REXP(34)
                  TEST A!0 = NUMBER
                     THEN A!1 := -A!1
                       OR A := LIST3(NEG, A, 0)
                  GOTO L

      CASE QNOT: NEXTSYMB()
                 A:=REXP(26)
                 A := LIST3(QNOT, A, 0)
                 GOTO L

      CASE QTABLE:NEXTSYMB()
                  A := REXP(11)
                  A := LIST3(QTABLE, A, 0)
                  GOTO L                          ]A


L:  SWITCHON SYMB INTO
    [B DEFAULT: RESULTIS A

      CASE RBRA:  NEXTSYMB()
                  B := 0
                  UNLESS SYMB=RKET DO
                                   [ B := REXP(0)
                                   UNLESS SYMB=RKET DO REPORT(9) ]
                  NEXTSYMB()
                  A := LIST3(FNAP, A, B)
                  GOTO L

      CASE VECAP:IF N GE 40 RESULTIS A
                 NEXTSYMB()
                 B := REXP(40)
                 A := LIST3(VECAP, A, B)
                 GOTO L

      CASE QREM:
      CASE MULT:
      CASE DIV: IF N GE 35 RESULTIS A
                OP := SYMB
                NEXTSYMB()
                B := REXP(35)
                A := LIST3(OP, A, B)
                GOTO L

      CASE LPLUS:
      CASE LMINUS:
      CASE PLUS:
      CASE MINUS:IF N GE 34 RESULTIS A
                 OP := SYMB
                 NEXTSYMB()
                 B := REXP(34)
                 A := LIST3(OP, A, B)
                 GOTO L

      CASE QLEQ: CASE QLNE: CASE QLGE: CASE QLGR: CASE QLLE:
```

```
        CASE QLLS:
        CASE QEQ:CASE QNE:
        CASE QLS:CASE QGR:
        CASE QLE:CASE QGE:
                  IF N GE 30 RESULTIS A
                   [ OP := SYMB
                     NEXTSYMB()
                     B := REXP(30)
                     A := LIST3(OP, A, B)
                     TEST C=0 THEN C := A
                              OR C := LIST3(QLOGAND, C, A)
                     SWITCHON SYMB INTO
                      [ DEFAULT: A := C
                                 GOTO L

                CASE QLEQ: CASE QLNE: CASE QLGR: CASE QLLS:
                CASE QLGE: CASE QLLE:
                        CASE QEQ:CASE QNE:CASE QLS:
                        CASE QGR:CASE QLE:CASE QGE:   A := 0  ]
                     A := B  ] REPEAT

CASE QLSHIFT:
CASE QASHIFT:
CASE QRSHIFT: IF N GE 32 RESULTIS A
                     OP := SYMB
                     NEXTSYMB()
                   B := REXP(32)
                     A := LIST3(OP, A, B)
                     GOTO L

CASE QLOGAND: IF N GE 24 RESULTIS A
                NEXTSYMB()
                B := REXP(23)
                A := LIST3(QLOGAND, A, B)
                GOTO L

CASE QLOGOR:IF N GE 23 RESULTIS A
              NEXTSYMB()
              B := REXP(22)
              A := LIST3(QLOGOR, A, B)
              GOTO L

CASE QEQV:
CASE QNEQV:IF N GE 22 RESULTIS A
             OP := SYMB
             NEXTSYMB()
             B := REXP(22)
             A := LIST3(OP, A, B)
             GOTO L

CASE COND:IF N GE 13 RESULTIS A
             NEXTSYMB()
             B := REXP(12)
             UNLESS SYMB=COMMA DO REPORT(10)
             NEXTSYMB()
             C := REXP(12)
```

```
                    A := LIST4(COND, A, B, C)
                    GOTO L

          CASE COMMA: IF N GE 12 RESULTIS A
                      NEXTSYMB()
                      B := REXP(11)
                      A := LIST3(COMMA, A, B)
                      GOTO L                ]B      ]1
//END       *END
//DECK CAE3
GET "CAEGD"
GET "MANFEST"
 LET RDEF() = VALOF
     [1 LET A, B, OP = 0, 0, 0
        LET N = RNAMELIST()
        SWITCHON SYMB INTO
         [ CASE RBRA:NEXTSYMB()
                    UNLESS N.0=NAME DO REPORT(12)
                    IF SYMB=NAME DO A := RNAMELIST()
                    UNLESS SYMB=RKET DO REPORT(13)
                    NEXTSYMB()
                    IF SYMB=QBE DO
                            [ NEXTSYMB()
                              B := RBLOCK()
                              A := LIST5(RTDEF, N, A, B, 0)
                              GOTO M   ]
                    IF SYMB=QEQ DO
                            [ NEXTSYMB()
                              B := REXP(0)
                              A := LIST5(FNDEF, N, A, B, 0)
                              GOTO M   ]
                    REPORT(14)
                    A := LIST3(VALDEF, N, LIST2(NUMBER, 0))
                    GOTO M

          DEFAULT: REPORT(16)

          CASE QEQ:  OP := VALDEF
                    NEXTSYMB()
                    IF SYMB=QVEC DO
                            [ OP := VECDEF
                              NEXTSYMB()
                              UNLESS N.0=NAME DO REPORT(15)   ]
                    A := REXP(0)
                    A := LIST3(OP, N, A)
                    GOTO M

          M:        IF SYMB=QAND DO
                            [ NEXTSYMB()
                              B := RDEF()
                              RESULTIS LIST3(QAND, A, B)   ]
                    RESULTIS A   ]1

AND RCOM() = VALOF
   [1 LET A, B, C, OP = 0, 0, 0, 0
```

```
SWITCHON SYMB INTO
 [  DEFAULT: RESULTIS 0

    CASE NAME:CASE NUMBER:CASE STRINGCONST:
    CASE QTRUE:CASE QFALSE:CASE QLV:CASE QRV:
    CASE RBRA:CASE QVALOF:CASE PLUS:CASE MINUS:
    // THESE ARE ALL THE SYMBOLS WHICH CAN START AN EXPRESSION
              A := REXP(0)
              IF SYMB=ASS DO  [     NEXTSYMB()
                                    B := REXP(0)
                                    A := LIST3(ASS, A, B)
                                    GOTO L  ]
              IF SYMB=COLON DO
                      [ UNLESS A.0=NAME DO REPORT(17)
                        NEXTSYMB()
                        B := RCOM()
                        A := LIST4(COLON, A, B, 0)
                        GOTO L  ]
              IF (A.0&AMASK)=FNAP DO
                      [ A.0 := A.0 -> FNAP +, RTAP
                        GOTO L  ]
              REPORT(18)
              GOTO L

    CASE QGOTO:
    CASE QRESULTIS: OP := SYMB
                    NEXTSYMB()
                    A := REXP(0)
                    A := LIST3(OP, A, 0)
                    GOTO L

    CASE QIF:
    CASE QUNLESS:
    CASE QWHILE:
    CASE QUNTIL: OP := SYMB
                 NEXTSYMB()
                 A := REXP(0)
                 UNLESS SYMB=QDO DO REPORT(19)
                 NEXTSYMB()
                 B := RCOM()
                 A := LIST3(OP, A, B)
                 GOTO L

    CASE QTEST: NEXTSYMB()
                A := REXP(0)
                UNLESS SYMB=QDO DO REPORT(20)
                NEXTSYMB()
                B := RCOM()
                UNLESS SYMB=QOR DO REPORT(21)
                NEXTSYMB()
                C := RCOM()
                A := LIST4(QTEST, A, B, C)
                GOTO L

    CASE QFOR: NEXTSYMB()
               TEST SYMB=NAME THEN A := RNAME()
```

```
                                OR  [ REPORT(28)
                                      NEXTSYMB()
                                      A := ERRORNAME  ]
            UNLESS SYMB=QEQ DO REPORT(22)
            NEXTSYMB()
          [ LET I, J = 0, 0
            I := REXP(0)
            UNLESS SYMB=QTO DO REPORT(23)
            NEXTSYMB()
            J := REXP(0)
            UNLESS SYMB=QDO DO REPORT(24)
            NEXTSYMB()
            B := RCOM()
            A := LIST5(QFOR, A, I, J, B)
            GOTO L  ]

CASE QBREAK:
CASE QFINISH:
CASE QRETURN: A := LIST1(SYMB)
              NEXTSYMB()
              GOTO L

CASE QSWITCHON: NEXTSYMB()
                A := REXP(0)
                UNLESS SYMB=QINTO DO REPORT(25)
                NEXTSYMB()
                B := RBLOCK()
                A := LIST3(QSWITCHON, A, B)
                GOTO L

   CASE QCASE: NEXTSYMB()
               A := REXP(0)
               UNLESS SYMB=COLON DO REPORT(26)
               NEXTSYMB()
               B := RCOM()
               A := LIST3(QCASE, A, B)
               GOTO L

   CASE QDEFAULT: NEXTSYMB()
                  UNLESS SYMB=COLON DO REPORT(27)
                  NEXTSYMB()
                  A := RCOM()
                  A := LIST3(QDEFAULT, A, 0)
                  GOTO L

   CASE SECTBRA: A := RBLOCK()
                 GOTO L  ]

L: SWITCHON SYMB INTO
    [ DEFAULT: RESULTIS A

      CASE QREPEAT: NEXTSYMB()
                    A := LIST3(QREPEAT, A, 0)
                    GOTO L

      CASE QREPEATWHILE:
```

```
            CASE QREPEATUNTIL: OP := SYMB
                               NEXTSYMB()
                               B := REXP(0)
                               A := LIST3(OP, A, B)
                               GOTO L  ]1
//END       *END
//DECK PLIST
GET "RCPLGD"
GET "MISCGD"
GET "MANFEST"

 LET PLIST(X, N, D) BE
       [1 LET SIZE = 0
          IF (X^AMASK)=0 DO  [ WRITES( "NIL" )
                             RETURN  ]
          SWITCHON X.0 ^ AMASK INTO
        [ CASE NUMBER: WRITES( "** NUMBER " )
                       WRITEN(X.1)
                       RETURN

          CASE NAME: WRITES( "** NAME " )
                     WRITEN(X^AMASK); WRITES("  ")
                     WRITES(X+3)
                     RETURN

          CASE STRINGCONST: WRITES( "** STRING " )
                            WRITES(X+1)
                            RETURN

          CASE QFOR: SIZE := SIZE + 1

          CASE COND:CASE FNDEF:CASE RTDEF:
          CASE QTEST:CASE CONSTDEF:
                    SIZE := SIZE + 1

          CASE LPLUS: CASE LMINUS: CASE QLEQ: CASE QLNE:
          CASE QLGR: CASE QLLS: CASE QLGE: CASE QLLE:
          CASE VECAP:CASE FNAP:
          CASE MULT:CASE DIV:CASE QREM:CASE PLUS:CASE MINUS:
          CASE QEQ:CASE QNE:CASE QGR:CASE QGE:CASE QLS:CASE QLE:
          CASE QASHIFT:
          CASE QLSHIFT:CASE QRSHIFT:CASE QLOGAND:CASE QLOGOR:
          CASE QEQV:CASE QNEQV:CASE COMMA:
          CASE QAND:CASE VALDEF:CASE VECDEF:
          CASE ASS:CASE RTAP:CASE COLON:CASE QIF:CASE QUNLESS:
          CASE QWHILE:CASE QUNTIL:CASE QREPEATWHILE:CASE QREPEATUNTIL:
          CASE QSWITCHON:CASE QCASE:CASE SEQ:CASE QLET:
          CASE QMANIFEST: CASE QSTATIC: CASE QGLOBAL: CASE QREPEAT:
          CASE QVALOF:CASE QLV:CASE QRV:CASE NEG:CASE QNOT:
          CASE QTABLE:
          CASE QGOTO:CASE QRESULTIS:
                    SIZE := SIZE + 2

          CASE QTRUE:CASE QFALSE:CASE QBREAK:CASE QRETURN:CASE QFINISH:
                    SIZE := SIZE + 1
```

```
                    IF N=0 DO  [ WRITES( *ETC* )
                               RETURN  ]

                    WRITES( *OP * )
                    WRITEN(X.0 ^ AMASK)
                    FOR I = 2 TO SIZE DO
                            [ WRITECH(OUTPUT, #*N#)
                              FOR J=0 TO N DO WRITES( *., *)
                              PLIST((SIZE=2->X.1,X.((I-1)/3)^
                                          -((I-1) MOD 3)*18),N+1,D) ]
                    RETURN   ]]

//END      *END
//DECK INPP1
GET *BCPLGD*
GET *CAEGD*
GET *MANFEST*

 LET NEXTSYMB() BE
  [1 IF B DO  [ SYMB, B -> NSYMB, FALSE
                RETURN  ]
     UNLESS CHKIND = EMPTY GOTO M

 NEXT: RCH()

 L: CHKIND := KIND(CH)
 M: SWITCHON CHKIND INTO
     [ CASE IGNORABLE: RCH() REPEATWHILE CH=#*S#
                       GOTO L

 CASE DIGIT: VP := 0
             RV V := 0
              [ VP := VP + 1
                V.(VP) := CH
        NUMB: RCH()
              CHKIND := KIND(CH)   ] REPEATWHILE CHKIND=DIGIT
             PP0(NUMBER, 1)
             IF CH=#B# THEN
                    [   CHKIND := EMPTY
                        TEST RV V=OCTAL THEN REPORT(29)
                                        ELSE RV V := OCTAL ]
             IF RV V=OCTAL THEN
                        FOR I=1 TO VP DO
                             UNLESS #0# LE V.I LE #7# DO
                                [ REPORT(29): RETURN ]
             RETURN

 CASE CAPITAL:
 CASE SMALL: VP := 0
               [ VP := VP + 1
                 V.(VP) := CH
                 RCH()
                 CHKIND:=KIND(CH)    ] REPEATWHILE CHKIND GE 4
             TEST VP=1 THEN PP1(NAME, 1)
                       OR LOOKUPWORD()
             RETURN ]
```

```
CHKIND ← EMPTY
SWITCHON CH INTO
 [      CASE ≠$≠: PP1(QRV,0); RETURN
       SECTB: CASE ≠[≠:
              [ VP ← 0
                  [ RCH()
                    CHKIND←KIND(CH)
                    UNLESS CHKIND GE 4 BREAK
                    VP ← VP+1
                    V.VP←CH ] REPEAT
                     PP2(SECTBRA,0)
                    RETURN ]
       SECTK: CASE ≠]≠:
                  [ VP←0
                    [ RCH()
                      CHKIND←KIND(CH)
                      UNLESS CHKIND GE 4 BREAK
                      VP←VP+1
                      V.VP←CH ] REPEAT
                      PP0(SECTKET,1)
                      RETURN ]
    CASE ≠+≠: CDOT(PLUS,LPLUS); RETURN
    CASE ≠=≠: CDOT(QEQ,QLEQ); RETURN
    CASE ≠≡≠: CDOT(QNE,QLNE); RETURN
    CASE ≠>≠: CDOT(QGR,QLGR); RETURN
    CASE ≠<≠: CDOT(QLS,QLLS); RETURN
    CASE ≠≤≠: PP0(QLOGOR,0); RETURN
    CASE ≠←≠: PP0(ASS,0); RETURN
    CASE ≠(≠: PP1(RBRA, 0); RETURN
    CASE ≠)≠: PP0(RKET, 1); RETURN
    CASE ≠,≠: PP0(COMMA, 0); RETURN
    CASE ≠;≠: PP0(SEMICOLON, 0); RETURN
    CASE ≠∧≠: PP0(QLOGAND, 0); RETURN
    CASE ≠∨≠: PP0(QNOT, 0); RETURN
    CASE ≠,≠: PP0(VECAP, 0); RETURN
    CASE ≠↑≠: PP0(QASHIFT,0); RETURN

    CASE ≠/≠: RCH()
                UNLESS CH = ≠/≠ DO  [  PP0(DIV, 0)
                                       CHKIND ← KIND(CH)
                                       RETURN  ]
                RCH() REPEATUNTIL CH=≠*N≠
    CASE ≠*N≠:  NLPENDING ← TRUE
                GOTO NEXT

    CASE ≠***≠:  PP0(MULT,0)
                  RETURN


    CASE ≠-≠:   RCH()
                TEST CH=≠>≠ THEN PP0(COND,0) OR
                TEST CH=≠.≠ THEN PP0(LMINUS,0) OR
                [ PP0(MINUS,0); CHKIND←KIND(CH)  ]
                RETURN
```

```
CASE ≠:≠:     RCH()
              IF CH = ≠=≠ DO  [  PP0(ASS, 0); RETURN  ]
              PP0(COLON, 0)
              CHKIND ← KIND(CH)
              RETURN

CASE ≠↓≠: CASE ≠*≠:
          [1 LET QUOTE = CH
             VP ← 0
       NSCH: RCH()
             IF CH = ≠*≠ DO
                     [ RCH()
                       VP ← VP + 1
                       V.(VP) ← CH = ≠T≠ -> ≠*T≠,
                                    CH = ≠S≠ -> ≠*S≠,
                                    CH = ≠N≠ -> ≠*N≠;
                  CH=≠0≠ -> VALOF[ LET X=0
                                   FOR I=1 TO 3 DO
                                   [ RCH()
                                     TEST ≠0≠ LE CH LE ≠7≠
                                       THEN X←(X ↑ 3)≤(CH=≠0≠)
                                       OR REPORT(29)  ]
                    RESULTIS X ],
                                   CH
                       GOTO NSCH  ]
             IF CH=≠*N≠ GOTO NSCH

              IF CH = QUOTE DO
                     [ LET T = STRINGCONST
                       IF CH=≠*≠ DO [ IF VP>8 DO REPORT(30)
                                      T ← CHARCONST  ]
                       PP0(T, 1)
                       RETURN   ]
                VP ← VP + 1
                IF VP=VMAX LOGOR CH=ENDOFSTREAMCH DO
                                   [ REPORT(32); GOTO NEXT  ]
                V.(VP) ← CH
                GOTO NSCH  ]1
      DEFAULT:  IF CH=ENDOFSTREAMCH THEN
                                   [ IF GETP=0 DO
                                             [ PP0(QEND,0)
                                               RETURN  ]
                                     ENDREAD(INPUT)
                                     GETP ← GETP - 1
                                     INPUT ← GETV.(GETP)
                                     NLPENDING ← TRUE
                                     LINEP ← 0
                                     GOTO NEXT  ]
                [ LET A = OUTPUT
                  OUTPUT ← MONITOR
                  REPORT(31); WRITEN(CH); WRITECH(OUTPUT, ≠/≠)
                  WRITECH(OUTPUT, CH); WRITECH(OUTPUT, ≠*N≠)
                  OUTPUT ← A
                  GOTO NEXT       ]1

AND CDOT(S,L) BE
```

```
    [1 IF CHKIND=EMPTY DO RCH()
       TEST CH=≠.≠
         THEN PP0(L,0)
         OR [ PP0(S,0); CHKIND↛KIND(CH) ]1

//END       *END
//DECK INPP2
GET ↓BCPLGD↓
GET ↓CAEGD↓
GET ↓MANFEST↓

 LET LOOKUPWORD() BE

[ LET Z= VEC 16
 [ LET K,I,J=0,1,0
   Z.0 ↛ 0
   UNTIL I>VP DO
   [ Z.J ↛ Z.J ↑ BYTEBITS ≤ V.I
     I ↛ I + 1
     K ↛ K+1
     IF K GE 8 DO [ J ↛ J+1; Z.J↛0 ; K ↛ 0 ] ] ]
   SWITCHON Z.0 INTO
  [ DEFAULT:
   NOTKEY: PP1(NAME,1); RETURN
     CASE ≠AND≠     : PP0(QAND,0     ); RETURN
     CASE ≠ASHIFT≠  : PP0(QASHIFT,0  ); RETURN
     CASE ≠BE≠      : PP0(QBE,0      ); RETURN
     CASE ≠BREAK≠   : PP2(QBREAK,2   ); RETURN
     CASE ≠CASE≠    : PP2(QCASE,0    ); RETURN
     CASE ≠DEFAULT≠: PP2(QDEFAULT,0); RETURN
     CASE ≠DO≠      : PP0(QDO,0      ); RETURN
     CASE ≠EQ≠      : PP0(QEQ,0      ); RETURN
     CASE ≠EQV≠     : PP0(QEQV,0     ); RETURN
     CASE ≠ELSE≠    : PP0(QOR,0      ); RETURN
     CASE ≠END≠     : PP0(QEND,0     ); RETURN
     CASE ≠EXTERNAL≠: TEST Z.1=≠≠
                      DO [ PP0(XTERNAL,0); RETURN ]
                      OR GOTO NOTKEY
     CASE ≠FALSE≠   : PP0(QFALSE,1   ); RETURN
     CASE ≠FOR≠     : PP2(QFOR,0     ); RETURN
     CASE ≠FINISH≠  : PP2(QFINISH,2  ); RETURN
     CASE ≠GOTO≠    : PP2(QGOTO,0    ); RETURN
     CASE ≠GE≠      : PP0(QGE,0      ); RETURN
     CASE ≠GET≠:      [  ST ↛ 0
                      NEXTSYMB()
                      UNLESS SYMB=STRINGCONST DO
                         [ REPORT(34)
                           RETURN  ]
                      UNTIL CH=≠*N≠ DO RCH()
                         [ LET S = VEC 5
                           V.(0) ↛ VP
                           PACKSTRING(V, S)
                           GETV.(GETP) ↛ INPUT
                           GETP ↛ GETP + 1
                           INPUT ↛ FINDINPUT(S)
                           NLPENDING ↛ TRUE
```

```
                        NEXTSYMB()
                        RETURN ] ]
CASE #GR#      : PP0(QGR,0      ); RETURN
CASE #GLOBAL#  : PP0(QGLOBAL,0  ); RETURN
CASE #IF#      : PP2(QIF,0      ); RETURN
CASE #INTO#    : PP0(QINTO,0    ); RETURN
CASE #LET#     : PP0(QLET,0     ); RETURN
CASE #LV#      : PP0(QLV,0      ); RETURN
CASE #LE#      : PP0(QLE,0      ); RETURN
CASE #LS#      : PP0(QLS,0      ); RETURN
CASE #LOGOR#   : PP0(QLOGOR,0   ); RETURN
CASE #LOGAND#  : PP0(QLOGAND,0  ); RETURN
CASE #LEQ#     : PP0(QLEQ,0     ); RETURN
CASE #LNE#     : PP0(QLNE,0     ); RETURN
CASE #LGR#     : PP0(QLGR,0     ); RETURN
CASE #LLS#     : PP0(QLLS,0     ); RETURN
CASE #LGE#     : PP0(QLGE,0     ); RETURN
CASE #LLE#     : PP0(QLLE,0     ); RETURN
CASE #LSHIFT#  : PP0(QLSHIFT,0  ); RETURN
CASE #MANIFEST#: TEST Z.1=##
                 DO [ PP0(QMANIFEST,0); RETURN ]
                 OR GOTO NOTKEY
CASE #MOD#     : PP0(QREM,0     ); RETURN
CASE #NE#      : PP0(QNE,0      ); RETURN
CASE #NEQV#    : PP0(QNEQV,0    ); RETURN
CASE #NOT#     : PP0(QNOT,0     ); RETURN
CASE #OR#      : PP0(QOR,0      ); RETURN
CASE #RETURN#  : PP2(QRETURN,2  ); RETURN
CASE #REM#     : PP0(QREM,0     ); RETURN
CASE #RSHIFT#  : PP0(QRSHIFT,0  ); RETURN
CASE #RV#      : PP1(QRV,0      ); RETURN
CASE #REPEAT#  : PP0(QREPEAT,2  ); RETURN
CASE #REPEATUN#: TEST Z.1=#TIL#
                 DO [ PP0(QREPEATUNTIL,0); RETURN ]
                 OR GOTO NOTKEY
CASE #REPEATWH#: TEST Z.1=#ILE#
                 DO [ PP0(QREPEATWHILE,0); RETURN ]
                 OR GOTO NOTKEY
CASE #RESULTIS#: TEST Z.1=##
                 DO [ PP2(QRESULTIS,0); RETURN ]
                 OR GOTO NOTKEY
CASE #SWITCHON#: TEST Z.1=##
                 DO [ PP2(QSWITCHON,0); RETURN ]
                 OR GOTO NOTKEY
CASE #TO#      : PP0(QTO,0      ); RETURN
CASE #TEST#    : PP2(QTEST,0    ); RETURN
CASE #TRUE#    : PP0(QTRUE,1    ); RETURN
CASE #THEN#    : PP0(QDO,0      ); RETURN
CASE #TABLE#   : PP0(QTABLE,0   ); RETURN
CASE #UNTIL#   : PP2(QUNTIL,0   ); RETURN
CASE #UNLESS#  : PP2(QUNLESS,0  ); RETURN
CASE #VEC#     : PP0(QVEC,0     ); RETURN
CASE #VALOF#   : PP0(QVALOF,0   ); RETURN
CASE #WHILE#   : PP2(QWHILE,0   ); RETURN
] ]
```

```
//END       *END
//DECK INPP3
GET ↓BCPLGD↓
GET ↓CAEGD↓
GET ↓MISCGD↓
GET ↓MANFEST↓


 LET RCH() BE
         [1 IF CH=*#N* THEN [2
            TEST SYNERROR = 0
                 THEN LINEP := 0
                 OR [3 FOR I = 1 TO LINEP DO
                                    WRITECH(MONITOR,LINEV.I)
                       LINEP := 0
                    [ LET OUT = OUTPUT
                      OUTPUT := MONITOR
                                    FOR J=1 TO SYNERROR MOD 100-1 DO
                              WRITECH(OUTPUT,*#S*)
                      WRITECH(OUTPUT,*#*N*)
                      WRITECH(OUTPUT,*#*N*)
                      WRITES(MESSAGE,(SYNERROR/100))
                      WRITECH(OUTPUT,*#*N*)
                      SYNERROR := 0
                      OUTPUT := OUT ]2
            READCH(INPUT, LV CH)
            IF LINEP<90 DO LINEP := LINEP + 1
            LINEV.LINEP := CH ]1

 AND INSERT(ITEM) BE
         [ NSYMB, B := SYMB, TRUE
           SYMB := ITEM  ]

 AND PP0(ITEM, FINALST) BE
         [ SYMB := ITEM
           NLPENDING := FALSE
           ST := FINALST  ]

 AND PP1(ITEM, FINALST) BE
         [ SYMB := ITEM

           IF ST NE 0 & NLPENDING DO INSERT(SEMICOLON)

           NLPENDING := FALSE
           ST := FINALST  ]

 AND PP2(ITEM, FINALST) BE
         [ SYMB := ITEM

           IF ST NE 0 DO
                  TEST NLPENDING
                       THEN INSERT(SEMICOLON)
                         OR IF ST=1 DO INSERT(QDO)

           NLPENDING := FALSE
```

```
            ST = FINALST  ]
 AND KIND(CH) = VALOF
     [   IF #A#  LE  CH  LE  #Z# RESULTIS CAPITAL
         IF #0#  LE  CH  LE  #9# RESULTIS DIGIT
         IF CH = #*S# | CH = #*T# | CH = 0 RESULTIS IGNORABLE
         RESULTIS SIMPLE ]

//END       *END
//DECK TRNGD
GLOBAL [   //    TRNGD     ( ALL TRN FILES USE THIS HEAD )
       //
            BREAKLABEL      :  35  //
            CASEP           :  36  //
            CASET           :  37  //
            CASETABLE       :  38
            COMCOUNT        :  39  //
            CURRENTBRANCH   :  40  //
            DEFAULTLABEL    :  41  //
            DVEC            :  42  //
            DVECE           :  43  //
            DVECP           :  44  //
            DVECS           :  45  //
            DVECT           :  46  //
            GLOBDECL        :  47  //
            GLOBDECLS       :  48  //
            GLOBDECLT       :  49  //
            RESULTLABEL     :  50  //
            SAVESPACESIZE   :  51  //
            SSP             :  52  //
            STATLABEL       :  53  //
            VECSSP          :  54  //
                             ]

EXTERNAL [
            ADDNAME
            ASSIGN
            CELLWITHNAME
            CHECKDISTINCT
            COMPDATALAB
            COMPENTRY
            COMPJUMP
            COMPLAB
            DECLLABELS
            DECLNAMES
            EVALCONST
            JUMPCOND
            LOAD
            LOADLIST
            LOADLV
            NEXTPARAM
            OUT1
            OUT2
            OUT2P
            OUT3P
            OUTC
```

```
          OUTL
          OUTN
          TRANS
          TRANSDEF
          TRANSREPORT
          TRANSSWITCH
         ]


//END       *END
//DECK TRNO
GET ↓BCPLGD↓
GET ↓TRNGD↓
GET ↓MISCGD↓
GET ↓MANFEST↓

 LET NEXTPARAM() = VALOF  [ PARAMNUMBER ← PARAMNUMBER + 1
                            RESULTIS PARAMNUMBER  ]

 AND TRANSREPORT(N, X) BE
         [ LET A = OUTPUT
           OUTPUT ← MONITOR
           REPORTCOUNT ← REPORTCOUNT + 1
           WRITECH(OUTPUT,≠*S≠) ; WRITES(MESSAGE,N)
           WRITES(↓   COMMANDS COMPILED ↓)
           WRITEN(COMCOUNT); WRITECH(OUTPUT, ≠*N≠)
           WRITECH(OUTPUT,≠*S≠)
           PLIST(X, 0, 4); WRITECH(OUTPUT, ≠*N≠)
           OUTPUT ← A  ]

LET COMPILEAE(X) BE
       [1 LET A = VEC 1400      // FOR DVEC
          LET D = VEC 100       // FOR GLOBDECL
          LET E = VEC 300      // FOR CASETABLE

          DVEC, DVECS, DVECE, DVECP, DVECT ← A, 3, 3, 3, 1400
          DVEC.(0), DVEC.(1), DVEC.(2)  ← 0, 0, 0

          GLOBDECL, GLOBDECLS, GLOBDECLT ← D, 0, 100

          CASETABLE, CASEP, CASET ← E, 0, 300

          RESULTLABEL, BREAKLABEL, DEFAULTLABEL ← 0, 0, 0

          COMCOUNT, CURRENTBRANCH ← 0, X

          MESSAGE ← TABLE
            ↓DEFAULT USED TWICE IN THE SAME SWITCH.↓,
            ↓COMPILER ERROR IN DECLNAMES.↓,
            ↓COMPILER ERROR IN DECLVARS.↓,
            ↓LEFT SIDE OF AN ASSIGNMENT STATEMENT IN ERROR.↓,
            ↓ERROR IN THE RIGHT HAND SIDE OF AN ASSIGNMENT.↓,
            ↓STRUCTURE WRONG IN AN EXPRESSION.↓,
            ↓ERROR IN OPERAND OF LV.↓,
            ↓UNDECLARED NAME:↓,
            ↓A FUNCTION OR ROUTINE HAS A DYNAMIC FREE VARIABLE.↓,
```

```
        *ERROR IN A CONSTANT EXPRESSION.*,
        *CONSTANT EXPRESSION ERROR: ILLEGAL OPERATOR.*,
        *NON-MANIFEST CONSTANT NAME IN CONSTANT EXPRESSION.*,
        *TOO MANY CASES IN A SWITCH.*,
        *TWO DATA ITEMS WITH THE SAME NAME AND SAME SCOPE.*,
        *TOO MANY NAMES DECLARED (>350).*,
        *TOO MANY INITIALIZED GLOBAL VARIABLE DEFINITIONS (>50).*,
        *UNKNOWN EXPRESSION OPERATOR.*,
        *EXPRESSION MISSING.*,
        *A BREAK COMMAND IS NOT INSIDE ANY LOOP.*,
        *A RESULTIS COMMAND IS NOT INSIDE A VALOF.*

    PARAMNUMBER := 0
    SAVESPACESIZE := 2
    SSP := 0
    OUT2(STACK, SSP)
    DECLLABELS(X)
    TRANS(X)
    OUT1(QFINISH)
    OUT2(QGLOBAL, GLOBDECLS/2)
  [ LET I = 0
    UNTIL I=GLOBDECLS DO
            [ OUTN(GLOBDECL.(I))
              OUTL(GLOBDECL.(I+1))
              I := I + 2  ]
    RETURN  ]
//END       *END
//DECK TRN1
GET *TRNGD*
GET *MANFEST*
 LET TRANS(X) BE
     [TRANS
        IF (X^AMASK)=0 RETURN
        CURRENTBRANCH := X
        SWITCHON X.0^AMASK INTO
         [ CASE QLET:  [ LET A, B, S = DVECE, DVECS, SSP
                       LET V = VECSSP
                       STATLABEL := 0
                       DECLNAMES(X.0↑-18)
                       UNLESS STATLABEL=0 DO COMPLAB(STATLABEL)
                       CHECKDISTINCT(B, DVECS)
                       DVECE := DVECS
                       VECSSP := SSP
                       SSP := S
                       TRANSDEF(X,0↑-18)
                       SSP := VECSSP
                       OUT2(STACK, SSP)
                       OUT1(STORE)
                       DECLLABELS(X,0↑-36)
                       TRANS(X.0↑-36)
                       VECSSP := V
                       DVECE, DVECS, SSP := A, B, S
                       OUT2(STACK, SSP)
                       RETURN  ]

           CASE QGLOBAL:
```

```
CASE XTERNAL:
CASE QMANIFEST:
        [1 LET A, B, S = DVECE, DVECS, SSP
           AND P=(X.0^AMASK)=QGLOBAL -> QGLOBAL;
                 (X.0^AMASK)=XTERNAL -> XTERNAL, NUMBER
            [ LET Y = X.0 ↑ -18 ∧ AMASK
              UNTIL Y=0 DO
                  [ ADDNAME(Y.0↑-36,P,(P=XTERNAL->Y,1,EVALCONST(Y,1)))
                    Y ↑ Y.0 ↑ -18 ∧ AMASK ]  ]
              CHECKDISTINCT(B, DVECS)
              DVECE ↑ DVECS
              DECLLABELS(X.0↑-36)
              TRANS(X.0↑-36)
              DVECE, DVECS, SSP ↑ A, B, S
              RETURN  ]1

CASE QSTATIC:
        [1 LET A, B, S = DVECE, DVECS, SSP
           LET L = NEXTPARAM()
           COMPJUMP(L)

            [ LET Y = X.0 ↑ -18
              UNTIL Y=0 DO
                    [  LET M = NEXTPARAM()
                       ADDNAME(Y.0↑-36, LABEL, M)
                       COMPDATALAB(M)
                       OUT2(ITEMN, EVALCONST(Y,1))
                       Y ↑ Y.0 ↑ -18 ]  ]
              CHECKDISTINCT(B, DVECS)
              DVECE ↑ DVECS
              DECLLABELS(X.0↑-36)
              COMPLAB(L)
              TRANS(X.0↑-36)
              DVECE, DVECS, SSP ↑ A, B, S
              RETURN   ]1

CASE ASS: ASSIGN(X.0↑-18, X.0↑-36)
          RETURN

CASE RTAP:
          [ LET S = SSP
            SSP ↑ SSP+SAVESPACESIZE
            OUT2(STACK, SSP)
            LOADLIST(X.0↑-36)
            LOAD(X.0↑-18)
            OUT2(RTAP, S)
            SSP ↑ S
            RETURN    ]

CASE QGOTO: LOAD(X.0↑-18)
            OUT1(QGOTO)
            SSP ↑ SSP - 1
            RETURN

CASE COLON: COMPLAB(X.1)
```

```
                    OUT2(STACK, SSP)
                    TRANS(X.0↑-36)
                    RETURN

        CASE QIF:
        CASE QUNLESS:  [ LET L = NEXTPARAM()
                         JUMPCOND(X.0↑-18,(X.0^AMASK)=QUNLESS,L)
                         TRANS(X.0↑-36)
                         COMPLAB(L)
                         RETURN  ]

        CASE QTEST:  [ LET L, M = NEXTPARAM(), NEXTPARAM()
                       JUMPCOND(X.0↑-18, FALSE, L)
                       TRANS(X.0↑-36)
                       COMPJUMP(M)
                       COMPLAB(L)
                       TRANS(X.1)
                       COMPLAB(M)
                       RETURN  ]

CASE QBREAK: TEST BREAKLABEL = 0
               DO TRANSREPORT(18)
               OR COMPJUMP(BREAKLABEL)
                  RETURN

        CASE QRETURN: OUT1(RTRN)
                      RETURN

        CASE QFINISH: OUT1(QFINISH)
                      RETURN

  CASE QRESULTIS: LOAD(X.0↑-18)
IF RESULTLABEL = 0 THEN TRANSREPORT(19)
                    OUT2P(RES, RESULTLABEL)
                    SSP ↓ SSP - 1
                    RETURN

  CASE QWHILE:
  CASE QUNTIL:
          [ LET L, M = NEXTPARAM(), NEXTPARAM()
            LET BL = BREAKLABEL
  BREAKLABEL ↓ NEXTPARAM()
            COMPJUMP(M)
            COMPLAB(L)
            TRANS(X.0↑-36)
            COMPLAB(M)
            JUMPCOND(X.0↑-18,(X.0^AMASK)=QWHILE, L)
 COMPLAB(BREAKLABEL) ; OUT2(STACK,SSP)
            BREAKLABEL ↓ BL
            RETURN  ]

  CASE QREPEAT:
  CASE QREPEATWHILE:
  CASE QREPEATUNTIL:
          [ LET L, BL = NEXTPARAM(), BREAKLABEL
  BREAKLABEL ↓ NEXTPARAM()
```

```
            COMPLAB(L)
            TRANS(X,0↑-18)
            TEST (X,0^AMASK)=QREPEAT
                      THEN COMPJUMP(L)
                 OR JUMPCOND(X,0↑-36,(X,0^AMASK)=QREPEATWHILE,L)
COMPLAB(BREAKLABEL) ; OUT2(STACK,SSP)
            BREAKLABEL ↑ BL
            RETURN  ]

 CASE QCASE:  [ LET L, Y = NEXTPARAM(), EVALCONST(X,0↑-18)
                IF CASEP GE CASET DO TRANSREPORT(12,X)
                CASETABLE,(CASEP) ↑ Y
                CASETABLE,(CASEP+1) ↑ L
                CASEP ↑ CASEP + 2
                COMPLAB(L)
                TRANS(X,0↑-36)
                RETURN  ]

 CASE QDEFAULT: UNLESS DEFAULTLABEL=0 DO TRANSREPORT(0,X)
                DEFAULTLABEL ↑ NEXTPARAM()
                COMPLAB(DEFAULTLABEL)
                TRANS(X,0↑-18)
                RETURN


 CASE QSWITCHON: TRANSSWITCH(X)
                 RETURN

 CASE QFOR:
          [ LET A, B = DVECE, DVECS
            LET L, M = NEXTPARAM(), NEXTPARAM()
            LET BL = BREAKLABEL
            LET S1 = SSP
BREAKLABEL ↑ NEXTPARAM()
            ADDNAME(X,0↑-18, LOCAL, S1)
            DVECE ↑ DVECS
            LOAD(X,0↑-36)
          [ LET S2 = SSP
            LOAD(X,1)
            OUT1(STORE)
            COMPJUMP(L)
            DECLLABELS(X,1↑-18)
            COMPLAB(M)
            TRANS(X,1↑-18)
            OUT2(LP, S1); OUT2(LN, 1); OUT1(PLUS); OUT2(SP, S1)
            COMPLAB(L)
            OUT2(LP, S1); OUT2(LP, S2); OUT1(QLE); OUT2P(JT, M)
   COMPLAB(BREAKLABEL)
            BREAKLABEL, SSP ↑ BL, S1
            OUT2(STACK, SSP)
            DVECE, DVECS ↑ A, B
            RETURN  ]  ]

CASE SEQ: TRANS(X,0↑-18)
          COMCOUNT ↑ COMCOUNT + 1
          TRANS(X,0↑-36)
```

```
                        RETURN                  ]TRANS
//END       *END
//DECK TRN2
GET ↓TRNGD↓
GET ↓MANFEST↓
 LET DECLNAMES(X) BE
    [DN IF (X^AMASK)=0 RETURN
        SWITCHON X.0 ^ AMASK INTO
         [ CASE QAND: DECLNAMES(X.0↑-18)
                      DECLNAMES(X.0↑-36)
                      RETURN

           CASE VECDEF:
           CASE VALDEF: DECLDYN(X.0↑-18)
                        RETURN

           CASE RTDEF:
           CASE FNDEF: X.1 ↑ (X.1 ^ AMASK) ≤ NEXTPARAM() ↑ 18
                       DECLSTAT(X.0↑-18,X.1↑-18)
                       RETURN

           DEFAULT:  TRANSREPORT(1,X)     ]DN


AND DECLDYN(X) BE
         [ IF (X^AMASK)=0 RETURN
           IF X.0=NAME DO
                       [ ADDNAME(X,LOCAL,SSP)
                         SSP ↑ SSP + 1
                         RETURN  ]
           IF (X.0^AMASK)=COMMA DO
                       [ ADDNAME(X.0↑-18,LOCAL,SSP)
                         SSP ↑ SSP + 1
                         DECLDYN(X.0↑-36)
                         RETURN  ]
           TRANSREPORT(2,X)  ]


AND DECLSTAT(X, L) BE
       [1 LET T = CELLWITHNAME(X)

          IF DVEC.(T+1) = XTERNAL RETURN

          IF DVEC.(T+1)=0GLOBAL DO
                       [ LET N = DVEC.(T+2)
                         ADDNAME(X, 0GLOBAL, N)
                         IF GLOBDECLS GE GLOBDECLT DO TRANSREPORT(15, X)
                         GLOBDECL.(GLOBDECLS) ↑ N
                         GLOBDECL.(GLOBDECLS+1) ↑ L
                         GLOBDECLS ↑ GLOBDECLS + 2
                         RETURN   ]

          IF STATLABEL=0 DO
                       [ STATLABEL ↑ NEXTPARAM()
                         COMPJUMP(STATLABEL)   ]
```

```
        [ LET M = NEXTPARAM()
          ADDNAME(X, LABEL, M)
          COMPDATALAB(M)
          OUT2P(ITEML, L)     ]1


AND DECLLABELS(X) BE
        [ LET B = DVECS
          STATLABEL := 0
          SCANLABELS(X)
          UNLESS STATLABEL=0 DO COMPLAB(STATLABEL)
          CHECKDISTINCT(B, DVECS)
          DVECE := DVECS    ]


AND CHECKDISTINCT(E, S) BE
            [ UNTIL E=S DO    [ LET P = E + 3
                                AND N = DVEC.(E)
                                WHILE P LS S DO
                                      [ IF DVEC.(P)=N DO
                                                TRANSREPORT(13,N)
                                        P := P + 3   ]
                                E := E + 3 ]  ]


AND ADDNAME(N, P, A) BE
            [ IF DVECS GE DVECT DO TRANSREPORT(14, N^AMASK)
              DVEC.(DVECS), DVEC.(DVECS+1), DVEC.(DVECS+2) := (N^AMASK), P, A
              DVECS := DVECS + 3   ]


AND CELLWITHNAME(N) = VALOF
                      [ LET X = DVECE
                        N := N ^ AMASK
                         [ X := X - 3
                           IF X=0 RESULTIS 0   ] REPEATUNTIL DVEC.(X)=N
                       RESULTIS X   ]


AND LOADLIST(X) BE
          [ IF (X^AMASK)=0 RETURN
            IF (X.0^AMASK)=COMMA DO  [ LOADLIST(X.0+-18)
                                       LOADLIST(X.0+-36)
                                       RETURN   ]
            LOAD(X)   ]


AND SCANLABELS(X) BE
       [ IF (X^AMASK)=0 RETURN
         SWITCHON X.0 ^ AMASK INTO
          [ DEFAULT:  RETURN

            CASE COLON: X.1 := NEXTPARAM()
                        DECLSTAT(X.0+-18, X.1)
                        SCANLABELS(X.0+-36)
                        RETURN
```

```
        CASE QIF:CASE QUNLESS:CASE QWHILE:CASE QUNTIL:
        CASE QSWITCHON:CASE QCASE:
                SCANLABELS(X.0↑-36)
                RETURN

        CASE QREPEATWHILE:CASE QREPEATUNTIL:
        CASE QDEFAULT:CASE QREPEAT:
                SCANLABELS(X.0↑-18)
                RETURN

        CASE SEQ: SCANLABELS(X.0↑-18)
                SCANLABELS(X.0↑-36)
                RETURN

        CASE QTEST:SCANLABELS(X.0↑-36)
                SCANLABELS(X.1)
                RETURN   ]  }


AND TRANSDEF(X) BE
    [1 IF (X^AMASK)=0 RETURN
       SWITCHON X.0 ^ AMASK INTO
        [ CASE QAND: TRANSDEF(X.0↑-18)
                TRANSDEF(X.0↑-36)
                RETURN

          CASE VECDEF: OUT2(LLP, VECSSP)
                SSP ↑ SSP + 1
                VECSSP ↑ VECSSP + 1 + EVALCONST(X.0↑-36)
                RETURN

   CASE VALDEF: LOADLIST(X.0↑-36)
                RETURN

   CASE FNDEF:
   CASE RTDEF:
          [2 LET L = NEXTPARAM()
             COMPJUMP(L)          // COMPILE JUMP ROUND BODY

          [3 LET A, B, C = DVECE, DVECS, DVECP
             LET S = SSP
AND BL,RL = BREAKLABEL,RESULTLABEL
BREAKLABEL,RESULTLABEL ↑ 0,0
             COMPENTRY(X.0↑-18,X.1↑-18) // THE ENTRY POINT
             SSP ↑ SAVESPACESIZE

             DVECP ↑ DVECS
             DECLDYN(X.0↑-36)    // DECLARE THE FPL
             CHECKDISTINCT(B, DVECS)
             DVECE ↑ DVECS
             DECLLABELS(X.1)     // DECLARE THE LABELS

             OUT2(SAVE, SSP)

             TEST (X.0^AMASK)=FNDEF
```

```
                                THEN  [ LOAD(X,1); OUT1(FNRN)  ]
                                  OR  [ TRANS(X,1); OUT1(RTRN)  ]
        BREAKLABEL,RESULTLABEL ← BL,RL
                    SSP ← S
                    OUT2(STACK, SSP)
                     OUT1(STORE)
                    DVECE, DVECS, DVECP ← A, B, C  ]3

                    COMPLAB(L)
                    RETURN  ]2

          DEFAULT: RETURN  ]1
//END       *END
//DECK TRN3
GET ↓TRNGD↓
GET ↓MANFEST↓
 LET JUMPCOND(X, B, L) BE
    [JC SWITCHON X.0 ^ AMASK INTO
        [ CASE QTRUE: IF B DO COMPJUMP(L)
                      RETURN

          CASE QFALSE: UNLESS B DO COMPJUMP(L)
                       RETURN

          CASE QNOT: JUMPCOND(X,0↑-18, NOT B, L)
                     RETURN

          CASE QLOGAND: TEST B THEN  [ LET M = NEXTPARAM()
                                       JUMPCOND(X,0↑-18, FALSE, M)
                                       JUMPCOND(X,0↑-36, TRUE, L)
                                       COMPLAB(M)  ]

                        OR     [ JUMPCOND(X,0↑-18, FALSE, L)
                                 JUMPCOND(X,0↑-36, FALSE, L)  ]
                RETURN

CASE QLOGOR:  TEST B THEN  [ JUMPCOND(X,0↑-18, TRUE, L)
                             JUMPCOND(X,0↑-36, TRUE, L)  ]

                      OR     [ LET M = NEXTPARAM()
                               JUMPCOND(X,0↑-18, TRUE, M)
                               JUMPCOND(X,0↑-36, FALSE, L)
                               COMPLAB(M)  ]
                RETURN

CASE COND:  [ LET M, N = NEXTPARAM(), NEXTPARAM()
              JUMPCOND(X,0↑-18, FALSE, M)
              JUMPCOND(X,0↑-36, B, L)
              COMPJUMP(N)
              COMPLAB(M)
              JUMPCOND(X,1, B, L)
              COMPLAB(N)
              RETURN  ]

DEFAULT: LOAD(X)
         OUT2P(B -> JT, JF, L)
```

```
                    SSP ← SSP - 1
                    RETURN      ]]C


//END       *END
//DECK TRN4
GET ↓TRNGD↓
GET ↓MANFEST↓

 LET  TRANSSWITCH(X) BE
       [1 LET P, DL = CASEP, DEFAULTLABEL
          LET L, M = NEXTPARAM(), NEXTPARAM()
          COMPJUMP(L)
          DEFAULTLABEL ← 0
          TRANS(X,0↑-36)
          COMPJUMP(M)
          COMPLAB(L)
          LOAD(X,0↑-18)
          IF DEFAULTLABEL=0 DO DEFAULTLABEL ← M
          OUT3P(QSWITCHON, (CASEP - P) / 2, DEFAULTLABEL)
        [ LET I = P
          UNTIL I = CASEP DO  [ OUTN(CASETABLE,(I))
                                OUTL(CASETABLE,(I+1))
                                I ← I + 2 ]
          SSP ← SSP - 1
          COMPLAB(M)
          CASEP, DEFAULTLABEL ← P, DL ]]
//END        *END
//DECK TRN5
GET ↓BCPLGD↓
GET ↓TRNGD↓
GET ↓MANFEST↓

 LET LOAD(X) BE
     [1 IF (X^AMASK) = 0 DO
                    [ TRANSREPORT(17, CURRENTBRANCH)
                      OUT2(LN, 0)
                      SSP ← SSP + 1
                      RETURN  ]

      [ LET OP = X.0 ^ AMASK
        SWITCHON OP INTO
         [ DEFAULT: TRANSREPORT(16, CURRENTBRANCH)
                    OUT2(LN, 0)
                    SSP ← SSP + 1
                    RETURN

           CASE DIV:CASE QREM:
           CASE MINUS:
           CASE QLS:CASE QGR:CASE QLE:CASE QGE:
           CASE QLSHIFT:CASE QRSHIFT:
           CASE QASHIFT: CASE LMINUS: CASE QLGR:
           CASE QLLS: CASE QLGE: CASE QLLE:
                    LOAD(X,0↑-18)
                    LOAD(X,0↑-36)
                    OUT1(OP)
```

```
            SSP := SSP - 1
            RETURN

CASE LPLUS: CASE QLEQ: CASE QLNE:
CASE VECAP:CASE MULT:CASE PLUS:CASE QEQ:CASE QNE:
CASE QLOGAND:CASE QLOGOR:CASE QEQV:CASE QNEQV:
          [ LET A, B = X.0↑-18, X.0↑-36
            IF A.0=NAME LOGOR A.0=NUMBER DO
                                A, B := X.0↑-36, X.0↑-18
            LOAD(A)
            LOAD(B)
            TEST OP=VECAP THEN  [ OUTI(PLUS); OUTI(QRV)    ]
                          OR OUTI(OP)
            SSP := SSP - 1
            RETURN  ]

CASE NEG:CASE QNOT:CASE QRV:
            LOAD(X.0↑-18)
            OUTI(OP)
            RETURN

CASE QTRUE:CASE QFALSE:
            OUTI(OP)
            SSP := SSP + 1
            RETURN

CASE QLV: LOADLV(X.0↑-18)
          RETURN

CASE NUMBER: OUT2(LN, X.1)
             SSP := SSP + 1
             RETURN

CASE STRINGCONST:  [  LET V = VEC 520
                      UNPACKSTRING(X+1, V)
                      OUT2(LSTR, V.(0))
                     WRITEVEC(OUTPUT,LV V.1,V.0)
                      SSP := SSP + 1
                      RETURN  ]

CASE NAME:
          [ LET T = CELLWITHNAME(X)
            LET K, N = DVEC.(T+1), DVEC.(T+2)
            IF T=0 DO TRANSREPORT(7,X)
            IF T LS DVECP ∧ K=LOCAL DO TRANSREPORT(8,X)
            SSP := SSP + 1
            SWITCHON K INTO
             [ DEFAULT:
               CASE NUMBER: OUT2(LN, N); RETURN

               CASE LOCAL:  OUT2(LP, N); RETURN

               CASE XTERNAL: OUTI(LX)
                             WRITEVEC(OUTPUT,N,3)
                             RETURN
```

```
                CASE QGLOBAL: OUT2(LG, N); RETURN
                CASE LABEL: OUT2P(LL, N); RETURN     ] ]

CASE QVALOF:    [ LET RL = RESULTLABEL
                  LET A, B = DVECS, DVECE
                  DECLLABELS(X.0↑-18)
                  RESULTLABEL ← NEXTPARAM()
                  OUT1(QVALOF)     // TELLS CG TO REMEMBER TP
                  TRANS(X.0↑-18)
                  COMPLAB(RESULTLABEL)
                  OUT2(RSTACK, SSP)
                  SSP ← SSP + 1
                  DVECS, DVECE ← A, B
                  RESULTLABEL ← RL
                  RETURN  ]

CASE FNAP:      [ LET S = SSP
                  SSP ← SSP + SAVESPACESIZE
                  OUT2(STACK, SSP)
                  LOADLIST(X.0↑-36)
                  LOAD(X.0↑-18)
                  OUT2(FNAP, S)
                  SSP ← S + 1
                  RETURN   ]

CASE COND:      [ LET L, M = NEXTPARAM(), NEXTPARAM()
                  LET S = SSP
                  JUMPCOND(X.0↑-18, FALSE, M)
                  LOAD(X.0↑-36)
                  COMPJUMP(L)
                  SSP ← S; OUT2(STACK, SSP)
                  COMPLAB(M)
                  LOAD(X.1)
                  COMPLAB(L)
                  RETURN   ]

CASE QTABLE:    [ LET L, M = NEXTPARAM(), NEXTPARAM()
                  COMPJUMP(L)
                  COMPDATALAB(M)
                  TAB(X.0↑-18)
                  COMPLAB(L)
                  OUT2P(LLL, M)
                  SSP ← SSP + 1 ]]

AND TAB(X) BE
[       LET N = (X.0^AMASK)=COMMA -> RV(X.0↑-18)^AMASK, X.0^AMASK
        AND H = (X.0^AMASK)=COMMA -> X.0↑-18, X
        TEST N = QTABLE
             THEN [     LET L = NEXTPARAM()
                        OUT2(ITEML, L)
                        IF (X.0^AMASK)=COMMA DO TAB(X.0↑-36)
                        COMPLAB(L)
                        TAB(H.0↑-18) ]
        OR TEST N = STRINGCONST
             THEN [     LET L = NEXTPARAM()
```

```
                                OUT2(ITEML, L)
                                IF (X.0^AMASK)=COMMA DO TAB(X.0↑-36)
                                COMPLAB(L)
                                [ LET V = VEC 200
                                  UNPACKSTRING(H+1,V)
                                  L ↑ V.0 ]
                                FOR I=1 TO L/8+1 DO OUT2(ITEMN,H,I) ]
          OR [                  OUT2(ITEMN, EVALCONST(H))
                                IF (X.0^AMASK)=COMMA DO TAB(X.0↑-36) ] ]

AND LOADLV(X) BE
    [1 IF (X^AMASK)=0 GOTO ERR

       SWITCHON X.0 ^ AMASK INTO
        [ DEFAULT: GOTO ERR

          CASE NAME:  [ LET T = CELLWITHNAME(X)
                        LET K, N = DVEC.(T+1), DVEC.(T+2)
                        IF T=0 DO TRANSREPORT(7, X)
                        IF T LS DVECP ^ K=LOCAL DO TRANSREPORT(8, X)
                        SSP ↑ SSP + 1
                        SWITCHON K INTO
                         [ CASE LOCAL:  OUT2(LLP, N); RETURN

                           CASE QGLOBAL: OUT2(LLG, N); RETURN

                           CASE LABEL:  OUT2P(LLL, N); RETURN    ]  ]

                ERR: TRANSREPORT(6, CURRENTBRANCH)
                     OUT2(LN, 0)
                     SSP ↑ SSP + 1
                     RETURN

             CASE QRV: LOAD(X.0↑-18)
                       RETURN


             CASE VECAP:  [ LET A, B = X.0↑-18, X.0↑-36
                            IF X.0=NAME DO
                                        A, B ↑ X.0↑-36, X.0↑-18
                            LOAD(A)
                            LOAD(B)
                            OUT1(PLUS)
                            SSP ↑ SSP - 1
                            RETURN     ]
                                          ]]
//END       *END
//DECK TRN6
GET ↓TRNGD↓
GET ↓MANFEST↓

 LET EVALCONST(X) = VALOF
       [ IF (X^AMASK)=0 DO
                     [ TRANSREPORT(9, CURRENTBRANCH)
                       RESULTIS 0  ]
```

```
    SWITCHON X.0 ^ AMASK INTO
     [ DEFAULT:  TRANSREPORT(10,X)
                 RESULTIS 0

       CASE NAME:  [ LET T = CELLWITHNAME(X)
                     IF DVEC.(T+1)=NUMBER RESULTIS DVEC.(T+2)
                     TRANSREPORT(11,X)
                     RESULTIS 0  ]

       CASE NUMBER: RESULTIS X.1

       CASE NEG: RESULTIS -EVALCONST(X.0↑-18)

CASE QLOGAND: RESULTIS EVALCONST(X.0↑-18) LOGAND EVALCONST(X.0↑-36)
CASE QLEQ:
CASE QEQ    : RESULTIS EVALCONST(X.0↑-18) EQ     EVALCONST(X.0↑-36)
CASE QLNE:
CASE QNE    : RESULTIS EVALCONST(X.0↑-18) NE     EVALCONST(X.0↑-36)
CASE QLGR:
CASE QGR    : RESULTIS EVALCONST(X.0↑-18) GR     EVALCONST(X.0↑-36)
CASE QLLS:
CASE QLS    : RESULTIS EVALCONST(X.0↑-18) LS     EVALCONST(X.0↑-36)
CASE QLGE:
CASE QGE    : RESULTIS EVALCONST(X.0↑-18) GE     EVALCONST(X.0↑-36)
CASE QLLE:
CASE QLE    : RESULTIS EVALCONST(X.0↑-18) LE     EVALCONST(X.0↑-36)
CASE QLSHIFT: RESULTIS EVALCONST(X.0↑-18) LSHIFT EVALCONST(X.0↑-36)
CASE QRSHIFT: RESULTIS EVALCONST(X.0↑-18) RSHIFT EVALCONST(X.0↑-36)
CASE QASHIFT: RESULTIS EVALCONST(X.0↑    -18) ↑  EVALCONST(X.0↑-36)
CASE QLOGOR : RESULTIS EVALCONST(X.0↑-18) LOGOR  EVALCONST(X.0↑-36)
CASE QEQV   : RESULTIS EVALCONST(X.0↑-18) EQV    EVALCONST(X.0↑-36)
CASE QNEQV  : RESULTIS EVALCONST(X.0↑-18) NEQV   EVALCONST(X.0↑-36)
CASE QNOT   : RESULTIS ¬ EVALCONST(X.0↑-18)
CASE COND   : RESULTIS EVALCONST(X.0↑-18) ->     EVALCONST(X.0↑-36),
                                                 EVALCONST(X.1)
CASE MULT   : RESULTIS EVALCONST(X.0↑-18) *      EVALCONST(X.0↑-36)
CASE DIV    : RESULTIS EVALCONST(X.0↑-18) /      EVALCONST(X.0↑-36)
CASE LPLUS  :
CASE PLUS   : RESULTIS EVALCONST(X.0↑-18) +      EVALCONST(X.0↑-36)
CASE LMINUS :
CASE MINUS  : RESULTIS EVALCONST(X.0↑-18) -      EVALCONST(X.0↑-36)  ] ]


AND ASSIGN(X, Y) BE
    [] IF (X^AMASK)=0 LOGOR (Y^AMASK)=0 DO
                [ UNLESS  X=0 ^ Y=0 DO TRANSREPORT(4, CURRENTBRANCH)
                  RETURN  ]

       SWITCHON X.0 ^ AMASK INTO
        [ CASE COMMA: UNLESS (Y.0^AMASK)=COMMA DO
                              [ TRANSREPORT(5, CURRENTBRANCH)
                                RETURN  ]
                      ASSIGN(X.0↑-18, Y.0↑-18)
                      ASSIGN(X.0↑-36, Y.0↑-36)
                      RETURN
```

```
        CASE NAME: [ LET T = CELLWITHNAME(X)
                     LET K, N = DVEC.(T+1), DVEC.(T+2)
                     IF T=0 DO TRANSREPORT(7, X)
                     IF T LS DVECP ^ K=LOCAL DO
                                     TRANSREPORT(8, X)
                     LOAD(Y)
                     SSP := SSP - 1
                     SWITCHON K INTO
                      [ CASE NUMBER: TRANSREPORT(8, X)
                                     N := 0

                        CASE LOCAL: OUT2(SP, N); RETURN

                        CASE GGLOBAL: OUT2(SG, N); RETURN

                        CASE LABEL: OUT2P(SL, N); RETURN  ]  ]

        CASE GRV:
        CASE VECAP: LOAD(Y)
                    LOADLV(X)
                    OUT1(STIND)
                    SSP := SSP - 2
                    RETURN

        DEFAULT: TRANSREPORT(3, CURRENTBRANCH)  ]1
//END     *END
//DECK TRN7
GET "BCPLGD"
GET "TRNGD"
GET "MISCGD"
GET "MANFEST"

LET COMPLAB(L) BE
          [ OUT2P(LAB, L)  ]

AND COMPENTRY(N, L) BE
        [ LET V = VEC 50
          UNPACKSTRING(N+3, V)
          OUT3P(ENTRY, V.(0), L)
          WRITEVEC(OUTPUT,LV V.1,V.0)  ]

AND COMPDATALAB(L) BE
          [ OUT2P(DATALAB, L)  ]

AND COMPJUMP(L) BE
          [ OUT2P(JUMP, L)  ]

AND OUT1(X) BE
      [ WRITEVEC(OUTPUT,LV X,1)  ]

AND OUT2(X, Y) BE
     [ WRITEVEC(OUTPUT,LV X,2)  ]

AND OUT2P(X, Y) BE
     [ WRITEVEC(OUTPUT,LV X,2)  ]
```

```
AND OUT3P(X, Y, Z) BE
    [ WRITEVEC(OUTPUT,LV X,3)  ]

AND OUTN(X) BE
       [ WRITEVEC(OUTPUT,LV X,1)  ]

AND OUTL(X) BE
       [ WRITEVEC(OUTPUT,LV X,1)  ]

AND OUTC(X) BE
       [ WRITEVEC(OUTPUT,LV X,1)  ]

//END      *END
//DECK OVERLAY
//  DECLARATIONS TO LINK WITH HEAD

MANIFEST [
          ALLOC = 0
          BEADCALL = 1
          SELFCALL = 2
          READ =3
          WRITE =4
          SENDE = 5
          GETE = 6
          FATHER = 7
          TEMP = 8
          THISCLIST = 10
          MYSCRATCH = 12
          MYCLASS = 13
          CBLK = 14
          PROBE = 15
          DELBLK = 16
          SPRETURN = 17
          REDSHP = 18
          DSPCLX = 19
          FRETRN = 20
          RESTORE = 21
          UDAT = 22
          CCLIST = 23
          CSPROC = 24
          CCC = 25
          CAPOUT = 26
          MKOPR = 27
          DELCL = 28
          MAPCHRW = 29
          MAPCHRO = 30
          DELSUB = 31
          SAVE = 32
          BCPL2CL = 33
          BCPL2SCR = 34
          BCPL2CLA = 35
          BCPL2S = 36
          BCPL2CALL = 37
]
```

```
//  DECLARATIONS TO LINK WITH HEAD

GLOBAL [
          FREECAP    :   1  // INDEX OF FIRST FREE CAPABILITY
          NCAPS      :   2  // NUMBER OF FREE ONES
    ]

EXTERNAL [
          XJ         // XJ(OPERATION,PAR1,...,PARN)
          XJF        // XJF(F-RET-LABEL,OPERATION,PAR1,...)
          BEAD       // BEAD(B6,B1,B7,X1,X2)
    ]

GLOBAL [ // TO LINK WITH FSP AND IO
          FLIST      :   5  // POINTER TO FREE LIST
          OUTPUT     :   6  // STANDARD OUTPUT STREAM
          C6TO7      :   7  // DISPLAY CODE ASCII CONVERSION VECTOR
          C7TO6      :   8  // ASCII DISPLAY CODE  CONVERSION VECTOR
          IOBASE     :   9  // OPEN STREAM LIST POINTER
          MONITOR    :  10
    ]

EXTERNAL [
          NEWVEC          // NEWVEC(SIZEWANTED)
          RETVEC          // RETVEC(V,SIZEOFV)
          PACKSTRING      // (VECTOR,STRING)
          UNPACKSTRING    // (STRING,VECTOR)
          INITIALIZEIO
          FINDINPUT       //   (*FNAME,USERNAME*)
          CREATEOUTPUT    //   (*FNAME,USERNAME*)
          READCH          // (STREAM,LV CH)
          WRITECH         // (STREAM,CH)
          WRITES          // OUTPUT*STREAM; WRITES(STRING)
          WRITEN          // OUTPUT*STREAM; WRITEN(NUMBER)
          WRITEO          // OUTPUT*STREAM; WRITEO(NUMBER)
          READN           // NUMBER READER N*READN(STREAM)
          TRACE
          TRACECALL
          TRACERETURN
          DUMP
          ENDREAD         // CLOSE INPUT STREAM
          ENDWRITE        // CLOSE OUTPUT STREAM
          IOERROR         // GENERAL IOERROR ROUTINE
          UNPACKTO        //   (STR,VEC,BREAKCH)
          WRITEVEC
          READVEC
          FLUSH           // FORCE OUT A LINE WITHOUT CR
          BCDWORD
          ASCII
          ENDOFSTREAM
          CLOSEALL
          ABORT
    ]

MANIFEST [ AMASK = 777777B  ; ENDOFSTREAMCH =144B
```

```
            ESCAPECH = 173B ]

MANIFEST[
            LOCATE = 0
            UPDATE = 1
            DELETE = 2
                     ]

GLOBAL[
            CREATED    : 14  ]

LET OVERLAY(BINARY,COMPASS,CHREENT,BNAME,CNAME,IDENT,ONAME) BE
[           LET ENTRY = VEC 3 AND CALLFILE = VEC 100
            IF CREATED GOTO CALL
            CREATED := TRUE
            BEAD(LOCATE,ENTRY,BCPL2S,BCDWORD(@BCPL2),BCDWORD(@S))
            BEAD(UPDATE,ENTRY,BCPL2S,BCDWORD(@BCPL2),BCDWORD(@S))
            XJ(READ,BCPL2S,0,CALLFILE,100)
            XJ(CCC,BCPL2CLA)
            XJ(CCLIST,ALLOC,BCPL2CL,CALLFILE,8)
            XJ(CSPROC,BCPL2CLA,FATHER,CALLFILE,4,CALLFILE,5+10,
                      CALLFILE,6,CALLFILE,7,BCPL2CL)
            [M          LET MAP, MAPNUM = CALLFILE+10, 0
            [           XJ(MAPCHRW+MAP,5,BCPL2CLA,MAPNUM,
                                VALOF[      IF MAP.0 =, 0 RESULTIS MYSCRATCH
                                            BEAD(LOCATE,ENTRY,TEMP,MAP,0,MAP,1)
                                            BEAD(UPDATE,ENTRY,TEMP,MAP,0,MAP,1)
                                            RESULTIS TEMP ],
(MAP,1=0->MAP,2+300,MAP,2),MAP,3,(MAP,4->(MAP,4^AMASK)=MAP,3,MAP,4))
                      MAP, MAPNUM := MAP + 6, MAPNUM + 1 ]
                                            REPEATUNTIL MAP,0
                      MAP, MAPNUM := MAP + 1, 14
                      UNTIL MAP,0 =, 0 DO
                      [           BEAD(LOCATE,ENTRY,TEMP,MAP,0,MAP,1)
                                  BEAD(UPDATE,ENTRY,TEMP,MAP,0,MAP,1)
                                  XJ(CAPOUT,BCPL2CL,MAPNUM,TEMP)
                                  MAP, MAPNUM := MAP + 2, MAPNUM + 1 ]M
            XJ(MKOPR,ALLOC,BCPL2CALL,0,BCPL2CLA,11)
            FOR I = 0 TO 10 DO XJ(UDAT,BCPL2CALL,I)
            XJ(CAPOUT,BCPL2CL,ALLOC,ALLOC)
            XJ(CAPOUT,BCPL2CL,BEADCALL,BEADCALL)
            XJ(CAPOUT,BCPL2CL,SELFCALL,BCPL2CALL)
            XJ(CAPOUT,BCPL2CL,READ,READ)
            XJ(CAPOUT,BCPL2CL,WRITE,WRITE)
            XJ(CAPOUT,BCPL2CL,SENDE,SENDE)
            XJ(CAPOUT,BCPL2CL,GETE,GETE)
            XJ(CAPOUT,BCPL2CL,FATHER,FATHER)
            XJ(CAPOUT,BCPL2CL,THISCLIST,BCPL2CL)
            XJ(CAPOUT,BCPL2CL,MYSCRATCH,MYSCRATCH)
            XJ(CAPOUT,BCPL2CL,MYCLASS,BCPL2CLA)
CALL:       XJ(BCPL2CALL,BINARY,COMPASS,CHREENT,BNAME,0,BNAME,1,
                      CNAME,0,CNAME,1,IDENT,0,IDENT,1,ONAME,0,ONAME,1)
            UNLESS CREATED RETURN
            CREATED := FALSE
            XJ(DELSUB,BCPL2CLA)
            XJ(DELCL,BCPL2CL) ]
```

```
//END
          IDENT     START2
          TITLE               B C P L   C O M P I L E R -- PASS 2
          TITLE                    SYSTEM LINK
          SPACE     10
          LIST      -R
GLBSIZE   EQU       100
STKSIZE   EQU       8000
FRECAPS   EQU       10
          SPACE     5
          EXT       GLOBAL0
          ORG       0
          SPACE     5
          BSSZ      2
          DATA      L.BCPL2.,L.CLASS.        CLASS CODE NAME
          DATA      3                   NO. MAP ENTRIES
          DATA      60                  SPACE FOR COMPILED MAP
          VFD       60/GLOBAL0+GLBSIZE+1+STKSIZE  FL
          VFD       60/ENTRY            ENTRY POINT
          VFD       60/NEXTCAP+FRECAPS
          VFD       60/GLBSIZE+STKSIZE+ENDCAPS    SCRH FILE SIZE
          TITLE               LOGICAL MAP ENTRIES
          SPACE     5
          DATA      0,0,0,0             FOR SYS COMMUNICATION
          VFD       60/ENDCAPS,60/0     LENGTH, R/W
          SPACE     5
          DATA      L.BCPL2.,L.S.          MAP ENTRY FOR CODE
          VFD       60/ENDCAPS          FILE ADDRESS
          VFD       60/ENDCAPS          CORE ADDRESS
          VFD       42/0,18/GLOBAL0-ENDCAPS+1     LENGTH
          DATA      1                   R/W
          SPACE     5
          DATA      0,0      SCRATCH FILE ENTRY
          VFD       60/ENDCAPS
          VFD       60/GLOBAL0
          VFD       60/GLBSIZE+STKSIZE      SIZE OF SC FILE
          DATA      0                   R/W
          SPACE     5
          DATA      -1                  END OF MAP SPECIFIERS
          TITLE                    CAPABILITIES
*
          SPACE     5
CAP14     DATA      L.CBLK.,L.OPERATE.
          DATA      L.PROBE.,L.OPERATE.
          DATA      L.DELBLK.,L.OPERATE.
SPRET     SET       *-CAP14
SPRET     SET       SPRET/2+14
          DATA      L.RETURN.,L.OPERATE.
          DATA      L.REDSHP.,L.OPERATE.
          DATA      L.DSPCLX.,L.OPERATE.
          DATA      L.FRETUR.,L.OPERATE.
ENDCAPS   DATA      0
NEXTCA    SET       ENDCAPS-CAP14
NEXTCAP   EQU       NEXTCA/2+14
          TITLE                    MACROS
```

```
          SPACE   5
*
*         MACROES TO LINK COMPASS WITH BCPL PROGRAMS
*
          SPACE   5
RESET     MACRO
          SB5     =XGLOBAL0
          SA2     B5
          SB6     X2
          SB1     1
          SB2     -1
          MX6     0
          ENDM
          SPACE   5
GLOBAL    MACRO   NAME,INDEX  TO INITIALIZE GLOBAL
          SX7     NAME
          SA7     B5+INDEX
          ENDM
          SPACE   5
SAVE      MACRO             PUT AT EACH ENTRY
          SA7     B6
          SX7     B6
          SA7     =XGLOBAL0
          ENDM
          SPACE   5
RETURN    MACRO
          RESET
          SA2     B6
          SB4     X2
          JP      B4
          ENDM
          TITLE                    COMPASS ROUTINES
*
*         REGISTER WHOMPING SUBROUTINES
*
          SPACE   5
          VFD     4/0,7/70B,7/52B,7/140B,35/0,60/0,60/0
XJ        SAVE
          SB4     B6+2       POINT AT CALL NAME
+         XJ      -4
-         VFD     30/0
          BX1     X6         IN CASE ANY RETURN
          RETURN
          SPACE   5
          VFD     4/0,7/70B,7/52B,7/46B,7/140B,28/0,60/0,60/0
XJF       SAVE
          SB4     B6+3
+         XJ      -4
-         VFD     30/1
+         SB4     1
-         JP      L1
+         SB4     B0
L1        BX1     X6
          RESET
          EQ      B4,B0,L2
          JP      B6+2       JUMP TO FRETURN
```

```
L2      SA2     B6          NORMAL RETURN
        SB4     X2
        JP      B4
        EJECT
        SPACE   5
        VFD     4/0,7/42B,7/45B,7/41B,7/44B,7/140B,21/0,60/0,60/0
BEAD    SAVE
        SA1     B6+3        DIR ENTRY OR STRING RES
        SB1     X1
        SA1     B6+4        CAPABILITY
        SB7     X1
        SA1     B6+5        NAME
        SA2     B6+6        USER NAME
        SA3     B6+2        ACTION
        SB6     X3
+       XJ      BEADLOC
-       VFD     30/0
        RETURN
BEADSC  EQU     1
BEADLOC VFD     60/BEADSC
        VFD     4/0,7/42B,7/45B,7/41B,7/44B,7/140B,21/0,60/0,60/0
        TITLE                       MASK TABLE
*
*       MASK  TABLE
*
        SPACE   5
        ENTRY   MASKTAB,ENTRY,XJ,XJF,BEAD
ONES    SET     60
MASKTAB BSS     0
        DUP     61
ZEROS   SET     60-ONES
        VFD     ZEROS/0,ONES/-0
ONES    SET     ONES-1
        ENDD
        TITLE                       ENTRY POINTS
*
        SPACE   5
*
*       CODE TO BE EXECUTED AT BEGINNING
*
        SPACE   5
ENTRY   SB5     GLOBAL0
        SX7     B5+GLBSIZE+1
        SA7     B5          HIDE INITIAL STACK P
        RESET
        SX7     =XXJRET
        GLOBAL  NEXTCAP,1
        GLOBAL  (NEXTCAP+FRECAPS),2
+       JP      =XSTART
        END
//DECK CGHEAD
MANIFEST  //  6400 OP-CODES
  [  PS=0; JPBPK=20B; ZRXK=30B; NZXK=31B; PLXK=32B; NGXK=33B
     EQBBK=40B; LTBBK=70B; BX=110B; LXK=200B; AXK=210B
     LXBX=220B; AXBX=230B; NXBX=240B; UXBX=260B; PXBX=270B
     IX=350B; DX=420B; MXK=430B; FXXDX=440B; NO=460B
```

```
    SA=500B; SB=600B; SX=700B
    X=-10B; XTX=0; XPX=10B; XMX=20B; MX=30B; MXTX=40B; MXPX=50B
    MXMX=60B; APK=0; BPK=10B; XPK=20B; XPB=30B; APB=40B
    AMB=50B; BPB=60B; BMB=70B  ]

GLOBAL   //   COMMON STORAGE CELLS
  [  CODE       : 15  // ASSEMBLED CODE
     ILC        : 16  //  INDEX OF CORE
     LITAB      : 17  //  LITERAL TABLE
     ISH        : 18  //  INSTRUCTION SHIFT
     RBIT       : 19  //  RELOCATION BIT TABLE
     RL         : 20  //  INDEX OF RBIT
     RBITSH     : 21  //  SHIFT FOR RBITS
     TP         : 22  //  STORED= STACK INDEX
     STP        : 23  //  UPPER STACK INDEX
     WH         : 24  //  STATUS OF STACK POSITION
     REL        : 25  //  INDEX OF X REGISTER
     CO         : 26  //  ADDITIVE CONSTANT
     TESTR      : 27  //  TEST REGISTER FOR SWITCHONS
     DEFLAB     : 28  //  DEFAULT LABEL
     LLAB       : 29  //  HIGHEST LABEL ENCOUNTERED FROM OCODE
     HLAB       : 30  //  LOWEST LABEL GENERATED BY CG
     OP         : 31  //  OCODE OPERATION
     SCREG      : 32  //  SCRATCH REGISTER FOR SWITCHES CG6
     LTI        : 33  //  LITERAL TABLE INDEX
     ENTAB      : 34  //  ENTRY TABLE
     ETI        : 35  //  ENTRY TABLE INDEX
     EXTAB      : 36  //  EXTERNAL TABLE
     EXTABI     : 37  //  EXTAB INDEX
     LEN        : 38  //  LAST EXTERNAL NAME ENTERED
     LABTAB     : 39  //  LABEL TABLE
     BINARY     : 40  //  TRUE IFF PRODUCING BINARY
     COMPASS    : 41  //  TRUE IFF PRODUCING LISTING
     INPUT      : 42  //  GLOBAL TO HOLD OCODE INPUT STREAM
     LASTJ      : 43  //  LAST JUMP LABEL
     ESTACK     : 44  //  ENTRY STACK
     ESP        : 45  //  ENTRY STACK POINTER
     CHREENT    : 46  //  TRUE IFF CHECKING RE-ENTRANCE OF CODE
     FUFLAG     : 47  //  FORCE UPPER FLAG
     SEQTEST    : 48  //  GENERATE CODE FOR SEQUENTIAL TEST - CG6
     JUMPSW     : 49  //  GENERATE CODE FOR JUMP SWITCH - CG6
     BINTEST    : 50  //  GENERATE CODE FOR BINARY SWITCH  CG6
     ]

MANIFEST   //  WH SIGNALS
  [ REG=0; CONST=1; STORED=2; UNDEF=3  ]

EXTERNAL //    SUBROUTINE ENTRY NAMES
  [  CG          // MAIN CODE GENERATOR ROUTINE
     READOP      //  READS OCODE OPERATIONS
     REEDN       // READS NUMBERS
     CLREG       //  CLEAR REGISTERS
     MAKEREG     //  GET OPERAND IN REGISTER
     HERE        // DUMP REMOTES
     NEWLAB      //  CREATE A NEW LABEL
     READL       //  READ OCODE LABEL
```

```
INS         // ASSEMBLE SHORT INSTRUCTION
INSK        // ASSEMBLE MX-LIKE INSTRUCTION
INSA        //  ASSEMBLE LONG INSTRUCTION
INSL        //  ASSEMBLE LONG INSTRUCTION WITH RELOCATION
INSX        //  ASSEMBLE LONG INSTRUCTION EXTERNAL
DECLAB      //  DECLARE A LABEL
REMOTEC     // REMOTE CONSTANT
REMOTES     //  REMOTES STRING
READREG     //  FIND A FREE X REGISTER
CJUMP       //  COMPILE CONDITIONAL JUMP
REGNOC      //  RESET STATUS OF STACK TOP
FETCH       // ASSEMBLE FETCH IF NECESSARY
MOVE        // MOVE FRM X REGISTER
PLORMI      // COMPILE PLUS OR MINUS
NEOREQ      // COMPILE NE OR EQ
LSORGR      // COMPILE LS, GR, LE, OR GE
LORRSHIFT   // COMPILE LLSHIFT OR RSHIFT
LOGOP       //  COMPILE LOGICAL OPERATION
CMULT       //  COMPILE MULTIPLICATION
UNPACKBITS  //  WORD TO VECTOR
GSWITCH     //  GENERATE SWITCH CODE
STEP        //  INCREMENT INSTRUCTION POSITION
SMALL       //  TEST IF LESS THAN 2**17
GENCON      // GENERATE A CONSTANT IN AN X REGISTER
DATA        // OUTPUT A DATA WORD
PSTORE      // PREPARE OPERAND TO BE STORED
FORCEUPPER  // FILL OUT INSTRUCTION WORD WITH NO S
CSWITCH     // COMPILE SWITCH   CG6
DECENTRY    // DECLARE ENTRY CG4
CASHIFT     // COMPILE ASHIFT  CG3
PINS        // PRINT INSTRUCTION - CG4
WRITEBIN    // SECOND PASS ASSEMBLER - CG4
CGREPORT    // DIAGNOSTIC PRINTER AND ABORTER
CDIV        // COMPILE DIVIDE  -  CG4
CREM        // COMPILE REMAINDER  -  CG4
CALL2       // SECOND HALF OF FNAP
CALL        // GENERATES CODE FOR FNAP AND RTAP
LLABEL      // GENERATES CODE FOR LL
ENTREE
VFD
]
MANIFEST  [     ///    TABLE SIZES
   MILC=6000B             // PROGRAM LENGTH
   MLTI=700               //  LITERAL TABLE
   METI=50                // ENTRIES
   MEXTI=400              // EXTERNALS
   MLAB=500               //  LABELS
     ]

//END       *END
//DECK CG0
GET ↓BCPLGD↓
GET ↓MANFEST↓
GET ↓CGHEAD↓

LET START() BE
```

```
[1 LET V = VEC 3000
   INITIALIZEIO(V,3000)
 [ LET V1 = VEC 20
   AND V2=VEC 20
   AND V3=VEC 20
   AND T1 = VEC MILC
   AND T2 = VEC MILC/15+2
   AND T3= VEC MLTI
   AND T4 = VEC METI
   AND T5 = VEC MEXTI
   AND T6 = VEC MLAB
   AND T7 = VEC 50
   AND BNAME = 9
   AND IDENT = 13
   AND CNAME = 11
   CHREENT ← RV 8
   INPUT  ← 15
   COMPASS ← RV 7
   BINARY ← RV 6
   // INITIALIZE TABLES AND INDEXES
   CODE,ILC,ISH,CODE.0 ← T1,0,45,0
   RBIT,RL,RBITSH,RBIT.0 ← T2+1,0,57,0
   LITAB,LTI ← T3,0
   ENTAB,ETI ← T4,0
   EXTAB,EXTABI,EXTAB.0 ← T5,1,0
   LEN ← 0
   LABTAB, LLAB,HLAB ← T6,0,MLAB
   FOR I=0 TO MLAB DO LABTAB,I ← -1
   WH,REL,CO ← V1,V2,V3
   ESTACK, ESTACK.0, ESP ← T7, 0, 1
   IF COMPASS DO
         OUTPUT ← CREATEOUTPUT(CNAME,0=0->0,CNAME)
   INPUT ← FINDINPUT(INPUT)
   MONITOR ← CREATEOUTPUT(0)
   IF COMPASS DO [ WRITES(↓*TIDENT*T↓)
                   WRITES(IDENT) ]

   CG()

   IF COMPASS [ WRITES(↓*N*TEND*N↓); ENDWRITE(OUTPUT) ]

   IF BINARY DO WRITEBIN(BNAME,BCDWORD(IDENT))

   OUTPUT ← MONITOR

   IF BINARY DO
   [      WRITES(↓*TPROGRAM LENGTH ↓)
          WRITECH(OUTPUT,(ILC ↑ -15 ∧ 7) + ≠0≠)
          WRITECH(OUTPUT,(ILC ↑ -12 ∧ 7) + ≠0≠)
          WRITECH(OUTPUT,(ILC ↑ - 9 ∧ 7) + ≠0≠)
          WRITECH(OUTPUT,(ILC ↑ - 6 ∧ 7) + ≠0≠)
          WRITECH(OUTPUT,(ILC ↑ - 3 ∧ 7) + ≠0≠)
          WRITECH(OUTPUT,(ILC ∧ 7) + ≠0≠)
          WRITES(↓ WORDS*N↓) ]
   CLOSEALL() ]1
//END      *END
```

```
//DECK CG1
GET "BCPLGD"
GET "MANFEST"
GET "CGHEAD"


LET CG() BE
  [A LET TPV = VEC 30
     AND GLOBLAB = NEWLAB()
     AND TPI = 0
     TP := -1
      STP := 0
     FORCEUPPER()
     INSL(EQBBK,0,0,GLOBLAB); FORCEUPPER()
   ESTACK,0 := GLOBLAB
      M: OP := READOP()
     M1:
         IF STP<0 [ OUTPUT := MONITOR
                    WRITES("  THE STACK INDEX HAS GONE NEGATIVE.*")
                    WRITES("  THERE IS A BUG IN THE COMPILER.*N")
                    ABORT()    ]
         SWITCHON OP INTO
   [ CASE STACK: [1 LET N=REEDN()-1
                    TEST N<TP THEN TP,STP:=N,0  OR
                    [ IF N>TP+STP
                         DO WH.(STP+1),REL.(STP+1):=UNDEF,N-TP
                      STP:=N-TP ]1; GOTO M

     CASE STORE: CLREG(0); TP,STP := TP+STP,0 ; GOTO M

FRAP:
 CASE RTAP:
 CASE FNAP: CALL(); GOTO M

CASE SAVE: TP:=REEDN()-1; STP:=0; INS(SA+BPB,7,6,0); GOTO M

 CASE FNRN: MAKEREG(STP,1)
 CASE RTRN: INS(SA+BPB,2,6,0)
            INS(SB+XPB,4,2,0)
            INSA(JPBBK,4,0,0)
            IF OP=FNRN DO STP:=STP-1
            HERE(); GOTO M

 CASE QVALOF:  TPI:= TPI+1; TPV.TPI:=TP; GOTO M

   CASE RES: MAKEREG(STP,1)
            INSL(EQBBK,0,0,READL())
            STP:=STP-1; HERE(); GOTO M

 CASE RSTACK: IF TPI LE 0 [ OUTPUT:= MONITOR
              WRITES("  TPI  LE  0.  THERE IS A BUG IN THE COMPILER.*N")
                        ABORT() ]
               TP  := TPV.TPI; TPI:=TPI-1; STP := REEDN()-TP
               FOR I=1 TO STP-1 DO WH.I := STORED
              WH.STP,REL.STP,CO.STP:=REG,1,0
              GOTO M
```

```
CASE JUMP: CLREG(0)
           LASTJ := READL()
           INSL(EQBBK,0,0,LASTJ)
           HERE(); GOTO M

CASE JT: CJUMP(TRUE); GOTO M

CASE JF: CJUMP(FALSE); GOTO M

CASE QGOTO: CLREG(0)
            INSA(JPBPK,6,0,TP+STP)
            STP:=STP-1; HERE(); GOTO M

CASE LP:
CASE LG: [ LET I=READREG()
           INSA(SA+BPK,I,(OP=LP->6,5),REEDN())
         REGNOC(I) ] ; GOTO M

CASE LL:  LLABEL(FRAP,M1); GOTO M
CASE LX: [ LET I=READREG() AND N=VEC 2
           N!0, N!1, N!2 := REEDN(), REEDN(), REEDN()
           OP := READOP()
           TEST OP=QGOTO
               THEN [ CLREG(0)
                      INSX(JPBPK,0,0,N)
                      HERE(); GOTO M ]
           OR TEST OP=FNAP <= OP = RTAP
               THEN [ LET S=REEDN() AND L=NEWLAB(); CLREG(0)
                      INSA(SB+BPK,6,6,S)
                      INSL(SX+BPK,7,0,L)
                      INSX(JPBPK,0,0,N)
                      STP := STP + 1
                      CALL2(L,S)
                      GOTO M ]
           OR [ LET I = READREG()
                INSX(SX+BPK,I,0,N)
                REGNOC(I); GOTO M1 ]

CASE LLP:
CASE LLG: [ LET I=READREG()
            INSA(SX+BPK,I,(OP=LLP->6,5),REEDN())
            REGNOC(I) ]; GOTO M

CASE LLL: [ LET I=READREG()
            INSL(SX+BPK,I,0,READL())
            REGNOC(I) ]; GOTO M

CASE LN: STP := STP+1
         WH,STP,CO,STP := CONST,REEDN()
         GOTO M

CASE LSTR: [ LET I=READREG()
             INSL(SX+BPK,I,0,REMOTES(REEDN()))
           REGNOC(I) ]; GOTO M
```

```
CASE QTRUE: STP→ STP+1
            WH.STP,CO.STP → CONST,TRUE; GOTO M

 CASE QFALSE: STP→STP+1
              WH.STP,CO.STP→CONST,0; GOTO M

 CASE QRV: FETCH(STP)
           [ LET I=READREG()
             TEST WH.STP=CONST
               DO INSA(SA+BPK,I,0,CO.STP)
               OR MOVE(SA,I,REL.STP,CO.STP)
             WH.STP,REL.STP,CO.STP → REG,I,0 ]
           GOTO M

 CASE SP:
CASE SG: [ LET I=PSTORE(STP)
           INSA(SA+BPK,I,(OP=SP->6,5),REEDN())
           STP→STP-1 ]; GOTO M

CASE STIND: [ LET I=PSTORE(STP-1)
              FETCH(STP)
              TEST WH.STP=CONST THEN INSA(SA+BPK,I,0,CO.STP)
              OR MOVE(SA,I,REL.STP,CO.STP)
              IF CHREENT DO INS(SA+AMB,5,I,5)  // TEST
              STP→STP-2 ]; GOTO M

 CASE LPLUS: MAKEREG(STP-1,0); MAKEREG(STP,0)
 CASE PLUS: PLORMI(TRUE,FALSE); GOTO M

 CASE LMINUS: MAKEREG(STP-1,0); MAKEREG(STP,0)
 CASE MINUS: PLORMI(FALSE,FALSE); GOTO M

 CASE QLEQ: MAKEREG(STP-1,0); MAKEREG(STP,0)
 CASE QEQ: GOTO NEOREQ(TRUE) -> M1, M

 CASE QLNE: MAKEREG(STP-1,0); MAKEREG(STP,0)
CASE QNE: GOTO NEOREQ(FALSE) -> M1, M

CASE QLLS: MAKEREG(STP-1,0); MAKEREG(STP,0)
CASE QLS: GOTO LSORGR(TRUE,FALSE) -> M1, M

CASE QLGE: MAKEREG(STP-1,0); MAKEREG(STP,0)
CASE QGE: GOTO LSORGR(TRUE,TRUE) -> M1, M

CASE QLGR: MAKEREG(STP-1,0); MAKEREG(STP,0)
CASE QGR: GOTO LSORGR(FALSE,FALSE) -> M1, M

 CASE QLLE: MAKEREG(STP-1,0); MAKEREG(STP,0)
CASE QLE: GOTO LSORGR(FALSE,TRUE) -> M1, M

 CASE QLSHIFT: LORRSHIFT(TRUE); GOTO M

 CASE QRSHIFT: LORRSHIFT(FALSE); GOTO M

 CASE QASHIFT: CASHIFT()     // SEE CG3
               GOTO M
```

```
CASE QLOGAND: LOGOP(XTX); GOTO M

CASE QLOGOR: LOGOP(XPX); GOTO M

CASE QEQV: LOGOP(MXMX); GOTO M

CASE QNEQV: LOGOP(XMX); GOTO M

CASE QNOT:
CASE NEG: FETCH(STP)
          CO.STP := -CO.STP
        UNLESS WH.STP=CONST DO INS(BX+MX,REL.STP,0,REL.STP)
        GOTO M

CASE MULT: FETCH(STP-1); FETCH(STP)
           TEST WH.(STP-1)=CONST^WH.STP=CONST
             THEN CO.(STP-1) := CO.(STP-1)*CO.STP
             OR
             TEST WH.STP=CONST THEN CMULT(STP-1,STP)
                               OR   CMULT(STP,STP-1)
             STP:=STP-1; GOTO M

CASE DIV: CDIV()     //  SEE CG4
          STP:=STP-1; GOTO M

CASE QREM: CREM()     //   SEE CG4
           STP:=STP-1; GOTO M
```

```
 CASE LAB: CLREG(0)
 CASE DATALAB: [ LET L=READL()
                 IF L>HLAB DO CGREPORT(*TOO MANY LABELS GENERATED.*)
                  IF L>LLAB DO LLAB:=L
                  DECLAB(L); GOTO M  ]

CASE ITEMN: DATA(REEDN()); GOTO M

CASE ITEML: INSA(SB+BPK,6,0,0)
            INSL(EQBBK,0,0,READL()); GOTO M
  CASE QSWITCHON: CSWITCH()     //  SEE CG6
                  GOTO M

CASE QFINISH:
                INSX(EQBBK,0,0,*XJRET*); GOTO M

 CASE QGLOBAL: [ LET N=REEDN()
                 DECLAB(GLOBLAB)
                 FOR I=1 TO N
                  DO [ LET G=0
                       IF COMPASS DO WRITES(*N***T*T*)
                       G := REEDN()
                       INSL(SX+BPK,7,0,READL())
                       INSA(SA+BPK,7,5,G)]
                 RETURN  ]

CASE ENTRY: ENTREE(); GOTO M
```

```
    DEFAULT: [ LET OUT = OUTPUT
               OUTPUT ← MONITOR
               WRITES(↓ UNKNOWN OCODE OPERATION: ↓)
               WRITEO(OP)
               WRITES(↓. THERE IS A BUG IN THE COMPILER.*N↓)
               ABORT() ]
     ]A
//END       *END
//DECK CG2
GET ↓MANFEST↓
GET ↓BCPLGD↓
GET ↓CGHEAD↓
LET CJUMP(B) BE
    [ CLREG(1)
      INSL((B=>NGXK,PLXK),0,MAKEREG(STP,0),READL())
      STP ← STP-1  ]

AND PLORMI(P, R) BE
    [ FETCH(STP-1); FETCH(STP)
      CO.(STP-1) ← CO.(STP-1) + (P => CO.STP, -CO.STP)
      IF R DO CO.(STP-1) ← -CO.(STP-1)
      TEST WH.STP=CONST
        THEN IF R ∧ WH.(STP-1)≡CONST DO
                INS(BX+MX,REL.(STP-1),0,REL.(STP))       OR
      TEST WH.(STP-1)=CONST
        THEN [ WH.(STP-1),REL.(STP-1) ← REG,REL.STP
               IF P EQV R DO INS(BX+MX,REL.STP,0,REL.STP)  ] OR
        TEST R
          THEN INS(IX+XMX,REL.(STP-1),REL.(STP),REL.(STP-1))
          OR INS(IX+(P=>XPX,XMX),REL.(STP-1),REL.(STP-1),REL.STP)
        STP ← STP-1  ]

AND NEEDREG(B) =
    VALOF [1 PLORMI(FALSE,WH.(STP-1)=CONST)
             [ LET I=MAKEREG(STP,0)
               OP ← READOP()
               TEST OP = JT
                 THEN [ CLREG(1)
                        INSL((B=>ZRXK,NZXK),0,I,READL())
                        STP←STP-1; RESULTIS FALSE ] OR
               TEST OP=JF
                 THEN [ CLREG(1)
                        INSL((B=>NZXK,ZRXK),0,I,READL())
                        STP←STP-1; RESULTIS FALSE ] OR
               [ LET L=NEWLAB()
                 INSL(ZRXK,0,I,L)
                 INS(MXK,I,0,60)
                 DECLAB(L)
                 IF B DO INS(BX+MX,I,0,I)
                 RESULTIS TRUE  ]1

AND LSORGR(B1,B2) =
    VALOF [1 LET BASE = B1 => STP-1,STP
             IF WH.BASE = CONST DO
               [   CO.BASE ← CO.BASE + 1
```

```
                B1, B2 → vB1, vB2 ]
        PLORMI(FALSE,vB1)
    [   LET I = MAKEREG(STP,0)
        OP → READOP()
        TEST OP = JT
          THEN [ CJUMP(-B2); RESULTIS FALSE  ] OR
        TEST OP = JF
          THEN [ CJUMP(B2); RESULTIS FALSE ] OR
        [ INS(AXK,I,0,59)
          IF B2 DO INS(BX+MX,I,0,I)
          RESULTIS TRUE  ]1

AND LORRSHIFT(B) BE
    [1 TEST WH.STP = CONST
       THEN TEST WH.(STP-1)=CONST
            THEN CO.(STP-1)→ B -> CO.(STP-1) LSHIFT CO.STP,
                                 CO.(STP-1) RSHIFT CO.STP
            OR [ LET I=MAKEREG(STP-1,0)
                 INS((B->LXK,AXK),I,0,CO.STP)
                 INS(MXK,0,0,(B->60-CO.STP,CO.STP))
                 INS(BX+(B->XTX,MXTX),I,I,0)  ]
       OR [2 LET I=MAKEREG(STP-1,0)
             FETCH(STP)
             MOVE(SB,4,REL.STP+CO.STP)
             TEST B
               THEN [ INSX(SA+BPK,REL.STP,4,↓MASKTAB↓)
                      INS(BX+XTX,I,REL.STP,I)
                      INS(LXBX,I,4,I)  ]
               OR [ INS(AXBX,I,4,I)
                    INSX(SA+BPK,REL.STP,4,↓MASKTAB↓)
                    INS(BX+XTX,I,REL.STP,I)  ]2
       STP → STP-1   ]1

AND LOGOP(O) BE
    [ TEST WH.(STP-1)=CONST ∧ WH.STP=CONST
      THEN  SWITCHON O INTO
          [ CASE XTX: CO.(STP-1)→CO.(STP-1) ∧ CO.STP; GOTO L
            CASE XPX: CO.(STP-1)→CO.(STP-1) ≤ CO.STP; GOTO L
            CASE MXMX: CO.(STP-1)→CO.(STP-1) EQV CO.STP; GOTO L
            CASE XMX: CO.(STP-1)→CO.(STP-1) NEQV CO.STP; GOTO L ]
      OR [ LET I=MAKEREG(STP-1,0)
           AND J=MAKEREG(STP,0)
           INS(BX+0,I,J,I)  ]
  L:  STP → STP-1  ]

AND CMULT(Y,C) BE
    [A IF WH.C ≡ CONST GOTO SMULT
       TEST CO.C=0
         THEN [ WH.(STP-1),CO.(STP-1)→ CONST,0 ] OR
       [2 LET V=VEC 60 AND J=1 AND I=MAKEREG(Y,0) AND K=0
          UNPACKBITS((CO.C LS 0 -> -CO.C,CO.C), V)
          IF V.0  GE  7 GOTO SMULT
          UNTIL V.J DO J→J+1
          UNLESS J=1 DO INS(LXK,I,0,J-1)
          K,J → J,J+1
          UNTIL V.J DO [ J→J+1; IF J GR 60 GOTO DONE ]
```

```
        INS(BX+X,0,I,0)
        [1 INS(LXK,0,0,J-K)
           INS(IX+XPX,I,I,0)
           K,J←J,J+1
           UNTIL V,J DO [ J←J+1; IF J GR 60 GOTO DONE ]1
        REPEAT
DONE:   IF CO.C LS 0 DO INS(BX+MX,I,0,I)
        WH.(STP-1),REL.(STP-1),CO.(STP-1)←REG,I,0  ]2
      RETURN
SMULT:  [ LET I=MAKEREG(STP-1,0)
          AND J=MAKEREG(STP,0)
          INS(PXBX,I,0,I); INS(PXBX,J,0,J)
          INS(DX+XTX,I,I,J); INS(UXBX,I,0,I)
               ]A

LET CALL() BE
    [   LET N=READN() AND T=MAKEREG(STP,0) AND L = NEWLAB()
        CLREG(1)
        INSA(SB+BPK,6,6,N)
        INS(SB+XPB,3,T,0)
        INSL(SX+BPK,7,0,L)
        INSA(JPBPK,3,0,0)
        CALL2(L,N) ]

LET CALL2(L,N) BE
    [   HERE()
        FORCEUPPER()
        VFD(15,STP+TP-N-2,TRUE)
        VFD(15,N,TRUE)
        VFD(30,VALOF [ [ TEST LABTAB,(ESTACK,(ESP-1) ^ 777777B)<0
                           THEN RESULTIS ESTACK,(ESP-1)↑-30
                           ELSE ESP ← ESP - 1] REPEAT]
                    ,ESTACK,(ESP-1)↑-30=0)
        DECLAB(L)
        INSA(SB+BPK,6,6,-N)
        TEST OP=FNAP THEN
             [    STP ← N - TP
                  WH.STP,REL.STP,CO.STP←REG,1,0 ;
                     ELSE STP← N - TP - 1 ]

LET LLABEL(FRAP,M1) BE
    [1  LET I=READREG()
        INSL(SA+BPK,I,0,READL())
        OP ← READOP()
        TEST OP=QGOTO
             THEN [   CLREG(0)
                      INS(SB+XPB,4,I,0)
                      INSA(JPBPK,4,0,0)
                      HERE()]
             OR TEST OP=FNAP ≤ OP=RTAP THEN [ REGNOC(I); GOTO FRAP ]
                  OR [ INS(SX+BPB,0,6,0)
                       INSK(LXK,0,30)
                       INS(BX+XPX,I,I,0)
                       REGNOC(I); GOTO M1 ]1

//END      *END
```

```
//DECK CG3
GET ↓BCPLGD↓
GET ↓MANFEST↓
GET ↓CGHEAD↓

LET READREG()=
    VALOF [ LET FREE = VEC 5
            FOR I=1 TO 5 DO FREE.I↦TRUE
            FOR I=1 TO STP DO
              TEST WH.I=REG THEN FREE.(REL.I)↦FALSE OR
              IF WH.I=UNDEF DO I↦REL.I
            FOR I=1 TO 5 DO
              IF FREE.I RESULTIS I
            [ LET I=1
              UNTIL WH.I=REG DO I ↦ (WH.I=UNDEF -> REL.I, I)+1
            MOVE(SX,7,REL.I,CO.I)
             INSA(SA+BPK,7,6,TP+I)
                    WH.I ↦ STORED
            RESULTIS REL.I  ]  ]

AND CLREG(J) BE
     [ FOR I=1 TO STP-J DO
         TEST WH.I=REG ≤ WH.I=CONST
                       THEN [ LET J=PSTORE(I)
                              INSA(SA+BPK,J,6,TP+I)
                         WH.I ↦ STORED ] OR
         IF WH.I=UNDEF DO I↦REL.I  ]

AND FETCH(I) BE
    [I IF WH.I=STORED DO
        [ LET J=READREG()
          TEST 0 LE TP + I LE 1
              THEN INS(SA+BPB,J,6,TP+I)
              OR  INSA(SA+BPK,J,6,TP+I)
          WH.I,REL.I,CO.I ↦ REG,J,0  ]]

AND MAKEREG(I,P) =
    VALOF [ FETCH(I)
            TEST WH.I=REG
              THEN [ IF P=0 DO P↦REL.I
                     MOVE(SX,P,REL.I,CO.I) ]
              OR [ IF P=0 DO P↦ READREG()
                   GENCON(P,CO.I)  ]
         WH.I,REL.I,CO.I ↦ REG,P,0
         RESULTIS P  ]

AND PSTORE(I)= WH.I=CONST ∧ CO.I=0 ∧ CO.I GE 0 -> 6, MAKEREG(I,7)

AND REGNOC(I) BE
    [ STP ↦ STP+1
      WH.STP,REL.STP,CO.STP ↦ REG,I,0  ]

AND MOVE(RT,I,J,C) BE
    [ TEST C=0
        THEN UNLESS I=J ∧ RT=SX DO
               TEST RT=SX THEN INS(BX+X,I,J,0)
```

```
            OR INS(RT+XPB,I,J,0) OR
      TEST C=1 THEN INS(RT+XPB,I,J,1) OR
      TEST C=-1 THEN INS(RT+XPB,I,J,2) OR
      INSA(RT+XPK,I,J,C)
        ]

AND GENCON(R,C) BE
    [1 TEST SMALL(C)
       THEN SWITCHON C INTO
    [ CASE 0: INS(BX+XMX,R,R,R); RETURN
      CASE 1: INS(SX+BPB,R,1,0); RETURN
      CASE -1: INS(SX+BPB,R,2,0) ; RETURN
       CASE 2: INS(SX+BPB,R,1,1); RETURN
      CASE-2: INS(SX+BPB,R,2,2); RETURN
       DEFAULT: INSA(SX+BPK,R,0,C); RETURN  ]
       OR [ LET V= VEC 60 AND I,J = 1,1
            UNPACKBITS(C,V)
        UNTIL V.I DO I+I+1
            J+I+1
            WHILE V.J ^ J  LE  60 DO J+J+1
            FOR K=J TO 60 DO IF V.K GOTO L
            INS(MXK,R,0,J-I)
            UNLESS J=61 DO INS(LXK,R,0,J-1)
            RETURN
         L: TEST 1 LE R LE 5 THEN INSL(SA+BPK,R,0,REMOTEC(C))
            OR [ LET I=READREG()
                 INSL(SA+BPK,I,0,REMOTEC(C))
                 INS(BX+X,R,I,0)  ]]

AND UNPACKBITS(W,V) BE
    [ V.0 +0
      FOR I=1 TO 60 DO
      [ V.I + (W ^ 1 LSHIFT I-1) ≡ 0
        IF V.I DO V.0 + V.0 +1  ]]


AND CASHIFT() BE
    [1 TEST WH.STP=CONST
         THEN TEST WH.(STP-1)=CONST
                THEN CO.(STP-1) + CO.(STP-1) + CO.STP
                OR [ LET I=MAKEREG(STP-1,0)
                     TEST CO.STP<0
                       THEN INS(AXK,I,0,-CO.STP)
                       OR INS(LXK,I,0,CO.STP)  ]
         OR [ LET I=MAKEREG(STP-1,0)
              FETCH(STP)
              MOVE(SB,4,REL.STP,CO.STP)
            INS(LXBX,I,4,I)  ]
      STP + STP-1  ]1

AND SMALL(N) = -(1↑17) <. N <. 1↑17 ^ (N LSHIFT 1)≡.-1

LET ENTREE() BE
       [  LET N=REEDN() AND X=READL()
          LET V=VEC 20 AND S=VEC 2
          S.0,S.1,S.2 + 0,0,0
```

```
        FOR I=1 TO N DO V.I ← REEDN()
        V.0 ← N
        PACKSTRING(V,S)
        FOR I=0 TO 2 DO DATA(S.I)
        DECLAB(X)
    POP: UNLESS LABTAB.(ESTACK.(ESP-1)) LE 0 DO[ ESP←ESP-1; GOTO POP]
        ESTACK.ESP,ESP,RV S←X↑30≤LASTJ,ESP+1,RV S ∧ #177B ≤ 140B
        DECENTRY(S) ]

//END       *END
//DECK CG4
GET ↓BCPLGD↓
GET ↓MANFEST↓
GET ↓CGHEAD↓

LET PINS(OP,I,J,K,FUNC) BE
[ LET A,B,C = 0,0,0
   WRITECH(OUTPUT,#*N#)
   IF FUFLAG DO [ WRITES(↓ ↓); FUFLAG←FALSE ]
  WRITECH(OUTPUT,#*T#)
  TEST SA LE OP<SB THEN [ A← ↓SA↓; GOTO L ]  OR
       TEST SB LE OP<SX THEN [ A← ↓SB↓; GOTO L ]  OR
       TEST SX LE OP    THEN [ A← ↓SX↓; GOTO L ]  OR
       GOTO M
 L:   SWITCHON OP∧77B INTO
   [
       CASE APK    :   B,C ← ↓*TA↓,↓+↓          ; GOTO N
       CASE BPK    :   B,C ← ↓*TB↓,↓+↓          ; GOTO N
       CASE XPK    :   B,C ← ↓*TX↓,↓+↓          ; GOTO N
       CASE XPB    :   B,C ← ↓*TX↓,↓+B↓         ; GOTO N
       CASE APB    :   B,C ← ↓*TA↓,↓+B↓         ; GOTO N
       CASE AMB    :   B,C ← ↓*TA↓,↓-B↓         ; GOTO N
       CASE BPB    :   B,C ← ↓*TB↓,↓+B↓         ; GOTO N
       CASE BMB    :   B,C ← ↓*TB↓,↓-B↓         ; GOTO N
   ]
M:   SWITCHON OP INTO  [
    CASE PS: WRITES(↓PS*N↓); RETURN
     CASE ZRXK   : A,B,C ← 0,↓ZR*TX↓,↓,↓      ; GOTO N
     CASE NZXK   : A,B,C ← 0,↓NZ*TX↓,↓,↓      ; GOTO N
     CASE PLXK   : A,B,C ← 0,↓PL*TX↓,↓,↓      ; GOTO N
     CASE NGXK   : A,B,C ← 0,↓NG*TX↓,↓,↓      ; GOTO N
     CASE EQBBK  : A,B,C ← ↓EQ*TB↓,↓,B↓,↓,↓   ; GOTO N
     CASE LTBBK  : A,B,C ← ↓LT*TB↓,↓,B↓,↓,↓   ; GOTO N
     CASE BX+X   : A,B,C ← ↓BX↓,↓+TX↓,0       ; GOTO N
     CASE BX+MX  : A,B,C ← ↓BX↓,0,↓*T-X↓      ; GOTO N
     CASE BX+XTX : A,B,C ← ↓BX↓,↓+TX↓,↓**X↓   ; GOTO N
     CASE BX+XPX : A,B,C ← ↓BX↓,↓+TX↓,↓+X↓    ; GOTO N
     CASE BX+XMX : A,B,C ← ↓BX↓,↓+TX↓,↓-X↓    ; GOTO N
     CASE BX+MXMX: A,B,C ← ↓BX↓,↓+T-X↓,↓-X↓   ; GOTO N
   CASE BX+MXTX: A,B,C ← ↓BX↓,↓+T-X↓,↓**X↓ ;
                [ LET T=J ; J,K ← K,T ]; GOTO N
     CASE JPBPK  : A,B,C ← ↓JP*TB↓,0,↓+↓       ; GOTO N
     CASE LXK    : A,B,C ← ↓LX↓,0,↓+T↓         ; GOTO N
     CASE AXK    : A,B,C ← ↓AX↓,0,↓*T↓         ; GOTO N
     CASE MXK    : A,B,C ← ↓MX↓,0,↓*T↓         ; GOTO N
     CASE LXBX   : A,B,C ← ↓LX↓,↓+TB↓,↓,X↓     ; GOTO N
```

```
      CASE AXBX   : A,B,C ← ↓AX↓,↓*TB↓,↓,X↓    ; GOTO N
      CASE NXBX   : A,B,C ← ↓NX↓,↓*TB↓,↓,X↓    ; GOTO N
      CASE PXBX   : A,B,C ← ↓PX↓,↓*TB↓,↓,X↓    ; GOTO N
      CASE UXBX   : A,B,C ← ↓UX↓,↓*TB↓,↓,X↓    ; GOTO N
      CASE IX+XPX : A,B,C ← ↓IX↓,↓*TX↓,↓+X↓    ; GOTO N
      CASE IX+XMX : A,B,C ← ↓IX↓,↓*TX↓,↓-X↓    ; GOTO N
      CASE DX+XTX : A,B,C ← ↓DX↓,↓*TX↓,↓*#X↓   ; GOTO N
      CASE FXXDX  : A,B,C ← ↓FX↓,↓*TX↓,↓/X↓    ; GOTO N
    ]
N! UNLESS A=0 DO [ WRITES(A); WRITECH(OUTPUT,I+#0#)   ]
   UNLESS B=0 DO [ WRITES(B); WRITECH(OUTPUT,J+#0#) ]
   UNLESS C=0 DO [ WRITES(C); FUNC(K) ] ]

AND CGREPORT(S) BE
  [ OUTPUT ← MONITOR
    WRITES(↓*N ERROR IN CODE GENERATION: ↓); WRITES(S)
    WRITECH(OUTPUT,#*N#); ABORT()   ]

AND WRITEBIN(FNAME,IDENT) BE
   [W LET F=CREATEOUTPUT(FNAME)
      AND V = VEC 17 AND COUNT = 0
      MANIFEST [ MA= AMASK LSHIFT 15; UA= AMASK LSHIFT 30 ]
      V.1 ← 77B↑54 ≤ 14↑36
      V.2 ← IDENT
      FOR J=3 TO 15 DO V.J ← 0
      FORCEUPPER()
      V.16 ← 34B↑54 ≤ 1↑36
      V.17← IDENT ≤ ILC      //  PROGRAM LENGTH
      WRITEVEC(F,V,18)       //  PREFIX AND PIDLE TABLES
      COUNT ← COUNT + 17
      UNLESS ETI=0 DO
    [ LET X= 36B↑54 ≤ ETI↑36
      COUNT ← COUNT + ETI + 1
      WRITEVEC(F,LV X,1)
      WRITEVEC(F,ENTAB,ETI)   ]
    [3 LET IC,IR = 0,0
       AND IC1,S = 0,-57
       UNTIL IC GE ILC DO
   [ [ LET R = RBIT.IR ↑ S ∧ 7
         TEST R=2
           DO CODE.IC ← CODE.IC∧vMA
                      ≤ LABTAB.((CODE.IC∧MA)↑-15)↑ 15 OR
         [ IF (R∧4)≡0 DO CODE.IC ← CODE.IC∧vUA
                         ≤ LABTAB.((CODE.IC∧UA) ↑ -30)↑ 30
           IF (R∧1)≡0 DO CODE.IC ← CODE.IC∧vAMASK
                         ≤ LABTAB.(CODE.IC ∧ AMASK) ]
       IC,S ← IC+1,S+4   ]
            REPEATUNTIL S GR 0
        [ LET T= IC GE ILC => ILC-IC1, 15
          RBIT.(IR-1)  ← 40B↑54 ≤ (T+1)↑36 ≤ 1↑18 ≤ IC1
          WRITEVEC(F,LV RBIT.(IR-1),2)
          WRITEVEC(F,LV CODE.IC1, T)
          COUNT ← COUNT + T + 2
          IR,IC1,S ← IR+1,IC,-57   ]3
   UNLESS EXTABI=1 DO
   [1 IF EXTABI<0 DO EXTABI ←-EXTABI+1
```

```
        [ LET X = 44B↑54 ≤ (EXTABI-1)↑36
          COUNT ↑ COUNT + EXTABI
          WRITEVEC(F,LV X,1)
          WRITEVEC(F,EXTAB+1,EXTABI-1)  ]1
        V.0 ↑ COUNT ↑ 18 ≤ 1 ↑ 59 ≤ 23B ↑ 36
        V.1 ↑ 60741033B ↑ 36
        WRITEVEC(F,V,2)
        V.0 ↑ F.168                          ///  CAPABILITY INDEX FOR STREAM
        V.1 ↑ INPUT.168                      ///  CAPABILITY INDEX FOR STREAM
        V.2 ↑ INPUT.164                      ///  PNAME INDEX FOR STREAM
        V.3 ↑ INPUT.165                      ///  UNAME INDEX FOR STREAM
        ENDWRITE(F)
        XJ(WRITE,V.0,0,LV COUNT,1)
        ENDREAD(INPUT)
        BEAD(0,V+5,V.1,V.2,V.3)
        BEAD(2,V+5,V.1,V.2,V.3)  ]W

AND CDIV() BE
    [ TEST WH.(STP-1)=CONST ∧ WH.STP=CONST
              THEN [ CO.(STP-1)↑CO.(STP-1)/CO.STP
                     RETURN  ] OR
           [2 LET A=MAKEREG(STP-1,0)
              UNLESS WH.STP=CONST GOTO SDIV
           [1 LET V= VEC 60
              UNPACKBITS(CO.STP,V)
              UNLESS V.0=1 GOTO SDIV
              [ LET J=1
                UNTIL V.J DO J↑J+1
                 INSK(AXK,A,J-1) ]1
           RETURN
      SDIV: [ LET B=MAKEREG(STP,0)
              INS(PXBX,A,0,A); INS(NXBX,A,0,A)
              INS(PXBX,B,0,B); INS(NXBX,B,0,B)
              INS(FXXDX,A,A,B)
              INS(UXBX,A,4,A); INS(LXBX,A,4,A)  ]
    ]

AND CREM() BE
    [ TEST WH.(STP-1)=CONST ∧ WH.STP=CONST
              THEN [ CO.(STP-1)↑CO.(STP-1) REM CO.STP
                     RETURN  ] OR
           [1 LET A=MAKEREG(STP-1,0)
              UNLESS WH.STP=CONST GOTO SREM
           [2  LET V= VEC 60
              UNPACKBITS(CO.STP,V)
              UNLESS V.0=1 GOTO SREM
              [ LET J=1
                UNTIL V.J DO J↑J+1
                INS(MXK,0,0,61-J) ]2
                INS(BX+MXTX,A,A,0)
           RETURN
      SREM: [ LET B=MAKEREG(STP,0)
              INS(PXBX,0,0,A); INS(NXBX,0,0,0)
              INS(PXBX,B,0,B); INS(NXBX,7,0,B)
              INS(FXXDX,0,0,7); INS(UXBX,0,4,0)
              INS(LXBX,0,4,0); INS(PXBX,0,0,0)
```

```
                    INS(DX,XTX,0,0,B); INS(UXBX,0,0,0)
                    INS(IX,XMX,A,A,0)
    ] ]
//END       *END
//DECK CG5
GET ↓BCPLGD↓
GET ↓MANFEST↓
GET ↓CGHEAD↓

LET READL() =
   VALOF [ LET X=0
           READVEC(INPUT,LV X,1)
           IF COMPASS DO [ WRITES(↓ L↓); WRITEN(X) ]
           RESULTIS X  ]

AND REEDN() =
   VALOF [ LET X=0
           READVEC(INPUT,LV X,1)
           IF COMPASS DO [ WRITECH(OUTPUT,≠ ≠); WRITEN(X) ]
           RESULTIS X  ]

AND READOP()=
   VALOF [ LET X=0
           READVEC(INPUT,LV X,1)
           IF COMPASS DO
           [ WRITES(↓*N***T↓); WRITES( X=STACK    -> ↓STACK↓,
                                       X=RTAP     -> ↓RTAP↓,
                                       X=FNAP     -> ↓FNAP↓,
                                       X=STORE    -> ↓STORE↓,
                                       X=SAVE     -> ↓SAVE↓,
                                       X=QVALOF   -> ↓QVALOF↓,
                                       X=RES      -> ↓RES↓,
                                       X=RSTACK   -> ↓RSTACK↓,
                                       X=JUMP     -> ↓JUMP↓,
                                       X=JT       -> ↓JT↓,
                                       X=JF       -> ↓JF↓,
                                       X=QGOTO    -> ↓QGOTO↓,
                                       X=LP       -> ↓LP↓,
                                       X=LG       -> ↓LG↓,
                                       X=LL       -> ↓LL↓,
                                       X=LX       -> ↓LX↓,
                                       X=LLP      -> ↓LLP↓,
                                       X=LLG      -> ↓LLG↓,
                                       X=LLL      -> ↓LLL↓,
                                       X=LN       -> ↓LN↓,
                                       X=LSTR     -> ↓LSTR↓,
                                       X=QTRUE    -> ↓QTRUE↓,
                                       X=QFALSE   -> ↓QFALSE↓,
                                       X=QRV      -> ↓QRV↓,
                                       X=SP       -> ↓SP↓,
                                       X=SG       -> ↓SG↓,
                                       X=STIND    -> ↓STIND↓,
                                       X=LPLUS    -> ↓LPLUS↓,
                                       X=PLUS     -> ↓PLUS↓,
                                       X=LMINUS   -> ↓LMINUS↓,
                                       X=MINUS    -> ↓MINUS↓,
```

```
                X=QLEQ    -> ↓QLEQ↓,
                X=QEQ     -> ↓QEQ↓,
                X=QLNE    -> ↓QLNE↓,
                X=QNE     -> ↓QNE↓,
                X=QLLS    -> ↓QLLS↓,
                X=QLS     -> ↓QLS↓,
                X=QLGE    -> ↓QLGE↓,
                X=QGE     -> ↓QGE↓,
                X=QLGR    -> ↓QLGR↓,
                X=QGR     -> ↓QGR↓,
                X=QLLE    -> ↓QLLE↓,
                X=QLE     -> ↓QLE↓,
                X=QLSHIFT -> ↓QLSHIFT↓,
                X=QRSHIFT -> ↓QRSHIFT↓,
                X=QASHIFT -> ↓QASHIFT↓,
                X=QLOGAND -> ↓QLOGAND↓,
                X=QLOGOR  -> ↓QLOGOR↓,
                X=QEQV    -> ↓QEQV↓,
                X=QNEQV   -> ↓QNEQV↓,
                X=QNOT    -> ↓QNOT↓,
                X=NEG     -> ↓NEG↓,
                X=MULT    -> ↓MULT↓,
                X=DIV     -> ↓DIV↓,
                X=QREM    -> ↓QREM↓,
                X=LAB     -> ↓LAB↓,
                X=DATALAB -> ↓DATALAB↓,
                X=ITEMN   -> ↓ITEMN↓,
                X=ITEML   -> ↓ITEML↓,
                X=QSWITCHON-> ↓QSWITCHON↓,
                X=QFINISH -> ↓QFINISH↓,
                X=QGLOBAL -> ↓QGLOBAL↓,
                X=ENTRY   -> ↓ENTRY↓,
                             ↓OTHER↓)
            WRITECH(OUTPUT,##T#) ]
        RESULTIS X  ]
AND STEP() BE       // INSTRUCTION INCREMENTER
    [1 ISH ← ISH-15
       IF ISH<0 DO
       [ ILC ← ILC + 1
         IF ILC>MILC DO CGREPORT(↓TOO MUCH GENERATED CODE (>6000B LOC)↓)
         CODE.ILC ← 0
         ISH ← 45
         RBITSH ← RBITSH-4
         IF RBITSH < 0 DO
         [ RL ← RL+1
           RBIT.RL ← 0
           RBITSH ← 57 ]]

AND FORCEUPPER() BE
    [ IF BINARY DO
      [1 UNTIL ISH=45 DO
         [ CODE.ILC ← CODE.ILC ≤ NO↑6 ↑ ISH; STEP() ]]
      IF COMPASS DO FUFLAG ← TRUE   ]

AND INS(OP,I,J,K) BE    // SHORT INSTRUCTION ASSEMBLER
    [ IF BINARY DO
```

```
      [ CODE.ILC ← CODE.ILC≤((OP≤I)↑6≤J↑3≤K)↑ISH
        STEP()  ]
      IF COMPASS DO PINS(OP,I,J,K,WRITEN)  ]

AND INSK(OP,I,K) BE [ INS(OP,I,0,K)  ]

AND INSA(OP,I,J,K) BE   // LONG INSTRUCTION ASSEMPLER
    [ IF BINARY DO
      [ IF ISH=0 DO [ CODE.ILC ← CODE.ILC ≤ NO↑6 ; STEP()  ]
        ISH ← ISH-15
        CODE.ILC ← CODE.ILC ≤ ((OP≤I)↑2I≤J↑I8≤K∧AM∧SK)↑ISH
        STEP()  ]
      IF COMPASS DO PINS(OP,I,J,K,WRITEN)  ]

AND INSL(OP,I,J,L) BE   // RELOCATABLE INSTRUCT. ASSEMBLER
    [ IF BINARY DO
      [ IF ISH=0 DO [ CODE.ILC ← CODE.ILC ≤ NO↑6 ; STEP()  ]
        ISH ← ISH-15
        CODE.ILC ← CODE.ILC ≤ ((OP≤I)↑2I≤J↑I8≤L)↑ISH
        RBIT.RL ← RBIT.RL ≤ 1↑(RBITSH*ISH/16)
        STEP()  ]
      IF COMPASS DO PINS(OP,I,J,L,WRITEL)  ]

AND INSX(OP,I,J,S) BE   // EXTERNAL REFERENCE ASSEMBLER
    [ S.0 ← (S.0 ∧ V177B) ≤ 140B
      IF BINARY DO
      [ IF ISH=0 DO [ CODE.ILC ← CODE.ILC ≤ NO↑6 ; STEP()  ]
        ISH ← ISH-15
        CODE.ILC ← CODE.ILC ≤ ((OP≤I)↑2I≤J↑I8)↑ISH
      [ LET S1 = BCDWORD(S)
        UNLESS S1=EXTAB.LEN DO
        [ IF EXTABI<0 DO EXTABI ← -EXTABI+1
         EXTAB.EXTABI ← S1
          LEN ← EXTABI
          EXTABI ← EXTABI+1  ]
        IF EXTABI>MEXTI DO
              CGREPORT(↓TOO MANY EXTERNAL REFERENCES(>400).↓)
        [ LET T= 1↑29 ≤ ISH/15↑27 ≤ 1↑18 ≤ ILC
        STEP()
          TEST EXTABI<0
            DO [ EXTAB.-EXTABI ← EXTAB.-EXTABI ≤ T
                 EXTABI ← -EXTABI+1 ]
            OR [ EXTAB.EXTABI ← T↑30
                 EXTABI ← -EXTABI ] ] ]
          ]
      IF COMPASS DO PINS(OP,I,J,S,WRITEX ) ]

AND DECLAB(N) BE   // LABEL DECLARER
    [
      [ IF LABTAB.N GE 0 DO CGREPORT(↓DUPLICATE LABEL DECLARATIONS.↓)
        FORCEUPPER()
        LABTAB.N ← ILC  ]
      IF COMPASS DO
        [ WRITES(↓*N L↓); WRITEN(N); WRITES(↓*TBSS*T0↓)
          FUFLAG ← FALSE  ] ]
```

```
AND REMOTEC(C) =    // CONSTANT GENRATOR
    VALOF [ LET L=NEWLAB()
            IF BINARY DO
            [ IF LTI>MLTI DO CGREPORT(*LITERAL TABLE OVERFLOW.*)
              LITAB.LTI := L*30 <= 1
              LITAB.(LTI+1) := C
              LTI := LTI+2  ]
            IF COMPASS DO
            [ WRITES(*N*TRMT*N L*); WRITEN(L)
              WRITES(*N*TDATA*T*); WRITED(C)
              WRITES(*B*N*TRMT*)  ]
            RESULTIS L  ]

AND REMOTES(N) =    //  STRING CONSTANT DUMPER
    VALOF [ LET L=NEWLAB()
            AND Z = LV LITAB.(LTI+1)
            AND V= VEC 128
            FOR I=1 TO N DO [ V.I := REEDN();
                              IF COMPASS DO
                              IF I MOD 8 = 0 DO
                                      WRITES(*N***T*T*) ]
            V.0 := N
            PACKSTRING(V,Z)
            N := N/BYTESPERWORD +1
            IF BINARY DO
            [ LITAB.LTI := L*30 <= N
              LTI := LTI + N + 1
              IF LTI>MLTI DO CGREPORT(*LITERAL TABLE OVERFLOW.*) ]
            IF COMPASS DO
            [ WRITES(*N*TRMT*N L*); WRITEN(L)
              WRITES(*T8SS*T0*N*)
              FOR I=Z TO Z+N-1 DO
                [ WRITES(*T*TDATA*T*); WRITED(RV I); WRITES(*B*N*) ]
              WRITES(*T*TRMT*)  ]
            RESULTIS L  ]

AND NEWLAB() =     //  NEW LABEL GENERATOR
    VALOF [ IF LLAB GE HLAB DO
                 CGREPORT(*TOO MANY LABELS GENERATED.*)
            HLAB := HLAB-1
            RESULTIS HLAB+1  ]

AND DATA(N) BE    //  DATA GENERATOR
    [ IF BINARY DO
      [ UNLESS ISH=45 DO [ ISH:=0; STEP()  ]
        CODE.ILC := N
        ISH := 0 ; STEP()  ]
      IF COMPASS DO
        [ WRITES(*N*TDATA*T*)
          WRITED(N)
          WRITECH(OUTPUT,'B') ] ]

AND VFD(BITS,DATUM,FUNC) BE
      [1 IF COMPASS DO
            [     WRITECH(OUTPUT,'*N')
                  IF FUFLAG DO WRITES(* *)
```

```
                FUFLAG ← FALSE
                WRITES(↓*TVFD*T↓); WRITEN(BITS)
                WRITECH(OUTPUT,≠/≠); (FUNC=>WRITEN,WRITEL)(DATUM) ]
        IF BINARY DO
           [    CODE.ILC←CODE.ILC≤(DATUM^((I↑BITS)-.1))↑(ISH-BITS+15)
                ISH ← ISH - BITS
                UNLESS FUNC DO
                     RBIT.RL ← RBIT.RL ≤ 1 ↑ (RBITSH+(ISH+15)/15)
                IF ISH  LE  0 DO STEP()  ]]

AND HERE() BE   // REMOTE DUMPER
    [ IF BINARY DO
      [ LET I=0
        UNTIL I GE LTI DO
        [ FORCEUPPER()
          LABTAB.(LITAB.I ↑ -30) ← ILC
          FOR J=1 TO LITAB.I^AMASK DO
          [ CODE.ILC ← LITAB.(I+J)
            ISH ← 0 ; STEP() ]
          I ← I+(LITAB.I^AMASK)+1 ]
        LTI ← 0 ]
      IF COMPASS DO WRITES(↓*N*THERE↓)  ]

AND DECENTRY(S) BE   //  ENTRY POINT DECLARER
    [ IF BINARY DO
      [ IF ETI>METI DO CGREPORT(↓TOO MANY ENTRIES↓)
        ENTAB.ETI ← BCDWORD(S)
        FORCEUPPER()
        ENTAB.(ETI+1) ← I↑18 ≤ ILC
        ETI ← ETI+2  ]
      IF COMPASS DO
      [ WRITES(↓*N*TENTRY*T↓); WRITES(S)
        WRITES(↓*N ↓); WRITES(S)
        WRITES(↓*TBSS*T0↓)
        FUFLAG ← FALSE  ]  ]

AND WRITEL(N) BE [ WRITES(↓L↓); WRITEN(N)  ]

AND WRITEX(S) BE [ WRITES(↓=X↓); WRITES(S)  ]
//END      *END
//DECK CG6
GET ↓BCPLGD↓
GET ↓MANFEST↓
GET ↓CGHEAD↓

 //  THIS DECK IS JUST FOR SWITCH GENERATION

LET CSWITCH() BE
    [ LET A= VEC 150
      AND L= VEC 150
      AND NC,LOW,UP = REEDN(),1,150
      DEFLAB ← READL()
      IF COMPASS DO WRITES(↓*N***T*T↓)
      FOR I=1 TO NC DO
      [ LET X=REEDN()
        TEST SMALL(X)
```

```
        THEN A.LOW,L.LOW,LOW ← X,READL(),LOW+1
        OR A.UP,L.UP,UP ← X,READL(),UP-1
      IF COMPASS DO WRITES(↓*N***T*T↓) ]
    SORT(A,L,LOW-1)
    SORT(LV A.UP,LV L.UP,150-UP)
    TESTR ← MAKEREG(STP,0)
    CLREG(1)
    SCREG ← READREG()
    [ LET BL = NEWLAB()
      INS(BX+X,0,TESTR,0); INS(AXK,0,0,17); INSL(NZXK,0,0,BL)
      SEQTEST,JUMPSW,BINTEST ← SSEQTEST,SJUMPSW,SBINTEST
    GSWITCH(A,L,LOW-1)
      DECLAB(BL)
      SEQTEST,JUMPSW,BINTEST ← LSEQTEST,LJUMPSW,LBINTEST
      GSWITCH(LV A.UP,LV L.UP,150-UP)   ]
    STP ← STP-1  ]

AND SORT(A,B,N) BE
    [1 LET T=0
       UNTIL N LE 1 DO
       [ LET LSW=1
         FOR I=1 TO N-1 DO
         IF A.I>A.(I+1) DO
         [ T,A.I,A.(I+1) ← A.I,A.(I+1),T
           T,B.I,B.(I+1) ← B.I,B.(I+1),T
           LSW ← I  ]
         N ← LSW ]1

AND GSWITCH(A,B,NCASES) BE
    [G TEST NCASES < 3
        THEN [ FOR I=1 TO NCASES DO SEQTEST(A.I,B.I)
               INSL(EQBBK,0,0,DEFLAB)  ] OR
      [ LET ONE=1 AND MIN=A.1 AND MAX=A.NCASES
      [ LET RANGE= MAX-MIN+ONE
        TEST 2*RANGE  LE  3*NCASES ∧ NCASES > 7
          THEN [1 JUMPSW(MIN,MAX)
               [ LET L=NEWLAB() AND J=1
                 INSL(JPBPK,4,0,L)
                 DECLAB(L)
                 FOR I=0 TO RANGE-1 DO
                 [ TEST A.J-MIN=I
                     THEN [ INSL(EQBBK,0,0,B.J)
                            J←J+1 REPEATUNTIL A.J-MIN > I ]
                     OR INSL(EQBBK,0,0,DEFLAB)
                   FORCEUPPER() ]1
          OR [ LET M = (NCASES+1)/2
               AND L=NEWLAB()
               BINTEST(A.M,B.M,L)
               GSWITCH(LV A.M,LV B.M,NCASES-M)
               DECLAB(L)
               GSWITCH(A,B,M-1)    ]G

AND SSEQTEST(A,L) BE
    [ MOVE(SB,4,TESTR,-A); INSL(EQBBK,4,0,L)  ]

AND SJUMPSW(I,A) BE
```

```
    [ MOVE(SB,4,TESTR,-I); INSL(LTBBK,4,0,DEFLAB)
      MOVE(SX,0,TESTR,-A-1); INSL(PLXK,0,0,DEFLAB)    ]

AND SBINTEST(A,L1,L2) BE
    [ MOVE(SB,4,TESTR,-A); INSL(EQBBK,4,0,L1)
      INSL(LTBBK,4,0,L2)   ]

AND LSEQTEST(A,L) BE
    [ GENCON(SCREG,A); INS(IX+XMX,SCREG,SCREG,TESTR)
    INSL(ZRXK,0,SCREG,L)   ]

AND LUUMPSW(I,A) BE
    [ GENCON(SCREG,I); INS(IX+XMX,SCREG,TESTR,SCREG)
      INSL(NGXK,0,SCREG,DEFLAB); INS(SB+XPB,4,SCREG,0)
      GENCON(SCREG,A); INS(IX+XMX,SCREG,SCREG,TESTR)
      INSL(NGXK,0,SCREG,DEFLAB)   ]

AND LBINTEST(A,L1,L2) BE
    [ GENCON(SCREG,A); INS(IX+XMX,SCREG,TESTR,SCREG)
      INSL(ZRXK,0,SCREG,L1); INSL(NGXK,0,SCREG,L2)   ]
//END      *END
```

END OF FILE