APL for the Burroughs B5500 by the Computer Science Group, University of Washington, modified by Burroughs Corporation.

Original document from Ed Vandergriff:

> "Originally this came from a long-ago colleague, George P. Williams (then of Huntsville, AL) who shared my interests in computer architecture and language implementation; if I recall correctly he encountered it as a student at Georgia Tech."

This scan produced by Paul Kimpel on 5-Sep-2013 with the following corrections needed:

With this document there are a few corrections, shown below as a table, where the scanner did not capture the entire page.

All of these occur at the very top of the indicated pages, which refer to the handwritten page numbers in the lower-right corners.

Page	Col	Text	Sequence
6	1?	COMMENT	00005615
34	21	GTR MAXSPROGS THEN %OFF THE END OF SP	03110920
36	5	BEGIN	03121255

38	9	END;
		INTEGER C;
			03140080
			03140081

56	9	IF RARG.SCALAR=0 THEN M:=M+RARG.RF;	03271020
86	5	SETFIELD(GTA,0,8,0); SETFIELD(GTA,8,8,0);	09228004

Original URL for this document: http://www.phkimpel.us/APL-B5500-Listing-19710111.pdf

```
09:18:38 THURSDAY, MAY 25, 1972
APL    /IMAGE
WORDS PER RECORD =    10. WORDS PER BLOCK = 150.  TOTAL RECORDS =      7274
CREATION DATE = 71069
DATE OF LAST ACCESS = 72146
```

```
BEGIN
% THIS APL/B5500 PROGRAM WAS DEVELOPED BY THE COMPUTER SCIENCE GROUP
% AT THE UNIVERSITY OF WASHINGTON UNDER THE SPONSORSHIP OF PROFESSOR
% HELLMUT GOLDE.  THE PROGRAM MAY NOT BE OFFERED FOR SALE OR LEASE
% IN ITS ORIGINAL OR ANY MODIFIED FORM.  ANY PUBLICATION RELATING TO
% THIS PROGRAM OR ANY MODIFICATION OF THE PROGRAM MUST EXPLICITLY CREDIT
% THE COMPUTER SCIENCE GROUP OF THE UNIVERSITY OF WASHINGTON AND THE
% PRINCIPAL IMPLEMENTORS, GARY KILDALL, LEROY SMITH, SALLY SWEDINE,
% AND MARY ZOSEL.  COMPUTER RESOURCES FOR THE DEVELOPMENT OF THE
% PROGRAM WERE MADE AVAILABLE BY THE UNIVERSITY OF WASHINGTON COMPUTER
% CENTER.
DEFINE VERSIONDATE="1-11-71"#;
%MODIFICATIONS FOR B-5500 TIME-SHARING MCP MADE BY:
% JOSE HERNANDEZ, BURROUGHS CORPORATION.
BOOLEAN BREAKFLAG;
ARRAY GTA[0:1];
LABEL FINIS; %GO THERE WHEN YOU ARE IN TROUBLE (SPOUT A MESSAGE)
BOOLEAN PROCEDURE LIBRARIAN(A,B); VALUE A,B; REAL A,B; FORWARD;
LABEL FAULTL; %FAULT LABEL
MONITOR EXPOVR,INTOVR,INDEX:=INDEXF,FLAG,ZERO;
REAL BIGGEST, NULLV;
INTEGER STACKSIZE,LIBSIZE;
     REAL STATUSWORD,CORELOC;
     BOOLEAN RETURN;
BOOLEAN MEMBUG,DEBUG;
COMMENT    MEMBUG  SWITCHES ---------------------
               BIT          FUNCTION           BIT      FUNCTION
-----------------------------------------------------------------------
                1                              25
                2                              26
                3                              27
                4                              28
                5        DUMP TYPES @ INSERT   30
                6       DUMP TYPES @ DELETE  30
                7                              31
                8                              32
                9                              33
               10                              34
               11                              35
               12                              36
               13                              37
               14                              38
               15                              39
               16                              40
               17                              41
               18                              42
               19                              43
               20        DUMP INDEX            44
               21                              45
               22        DUMP TYPES            46
               23        CHECK TYPES           47
               24        DUMP BUFFER #S
                ;
FILE PRINT 4 "SYSTEMS" " BOX    " (1,15);
FILE TWXIN 19(2,30),TWXOUT 19(2,10);
%
DEFINE
     PAGESIZE=120#,
     AREASIZE=40#,
     CF=[26:13]#,       COMMENT COUNT FIELD -- NUMBER OF ITEMS ON PAGE;
     TF=[39:9] #,       COMMENT     T-FIELD (TYPE FIELD);
     FF=[9:1]#,     COMMENT FULL FIELD FOR SEQUENTIAL STORAGE;
     AF=[1:23] #,       COMMENT A-FIELD;
     BF=[24:23]#,       COMMENT B-FIELD;
     MF=[1:1]#,       COMMENT METHOD OF STORAGE FIELD;
     SF=[13:13]#,       COMMENT SEQUENTIAL STORAGE SIZE FIELD (#CHRS);
     BOOL=[47:1]#,
     SKIP=1#,           COMMENT --AMOUNT OF SPACE RESERVED AT THE
                        START OF EACH PAGE;
     ALLOWANCE=10#,     COMMENT --DEVIATION FROM THE AVERAGE PAGE SIZE
```

```
                        ALLOWED BEFORE CORRECTION;
    RECSIZE=2#,
    MAXPAGES=20#,
    PAGESPACE=20#,
    NEXTP=[42:6]#,
    LASTP=[36:6]#,
    PAGEF=[19:11]#,
    BUFF=[12:6]#,
    CHANGEDBIT=[1:1]#,
    MBUFF=8#,
    SBUFF=4#,
    FLAGB=[18:1]#,     COMMENT FLAG BIT FOR BUFFER MAINTENANCE;
    EXTRAROOM=1#,
    LIBJOB="/APLIBE"#,%MFID FOR APL SYSTEM FILE
    ENDOFDEFINES=#;
REAL PROCEDURE CDR(X); VALUE X; REAL X; CDR:=X.NEXTP;
PROCEDURE RPLACD(X,Y); VALUE Y;REAL X,Y; X.NEXTP:=Y;
BOOLEAN PROCEDURE NULL(X); VALUE X; REAL X;  NULL:=X.NEXTP=0;
BOOLEAN STREAM PROCEDURE EOFMARK(SK,RS,A); VALUE SK,RS;
    BEGIN LABEL NO; SI:=A; SK(SI:=SI+8);
    RS(8(  2(IF SB THEN JUMP OUT 3 TO NO; SKIP SB));
    3(IF SB THEN SKIP SB ELSE JUMP OUT 3 TO NO); IF SB THEN
    JUMP OUT 2 TO NO; SKIP SB));TALLY:=1;EOFMARK:=TALLY;
    NO:
    END;
STREAM PROCEDURE MARKEOF(SK,RS,A); VALUE SK,RS;
    BEGIN DI:=A;
    SK(DI:=DI+8);
    RS(8(DS:=2RESET; DS:=3SET; DS:=RESET));
    END;
SAVE FILE ESTABLISH DISK [MAXPAGES:AREASIZE]
    (1,PAGESIZE,SAVE 100);
FILE NEWDISK DISK (1,PAGESIZE);
FILE DISK1 DISK (1,PAGESIZE),
     DISK2 DISK (1,PAGESIZE),
     DISK3 DISK (1,PAGESIZE),
     DISK4 DISK (1,PAGESIZE),
     DISK5 DISK (1,PAGESIZE),
     DISK6 DISK (1,PAGESIZE),
     DISK7 DISK (1,PAGESIZE),
     DISK8 DISK (1,PAGESIZE);
SWITCH FILE POINTERS:=DISK1,DISK1,DISK2,DISK3,DISK4,DISK5,DISK6,DISK7,
    DISK8;
PROCEDURE SETPOINTERNAMES;
    BEGIN
        IF NOT LIBRARIAN(LIBJOB,TIME(-1)) THEN
        BEGIN
        WRITE(ESTABLISH);
        MARKEOF(SKIP,RECSIZE,ESTABLISH(0));
        WRITE(ESTABLISH[1]);
        WRITE(ESTABLISH[MAXPAGES×AREASIZE-1]);
        LOCK(ESTABLISH);
        CLOSE(ESTABLISH)
        ;LIBSIZE←-1;
        END
    END;
DEFINE
    LIBMAINTENANCE=0#,
    MESSDUM=#;
    PROCEDURE MEMORY(MODE,TYPE,A,N,M);VALUE MODE,TYPE;
        INTEGER MODE,TYPE,N,M; ARRAY A[0]; FORWARD;
STREAM PROCEDURE MOVE(A,N,B); VALUE N;
    BEGIN SI:=A; DI:=B; DS:=N WDS;
    END;
PROCEDURE MESSAGE(I); VALUE I; INTEGER I;
    BEGIN
    FORMAT F("MEMORY ERROR",I5);
COMMENT CHANGE LINE 3050 TO WRITE(PRINT,SF[I]) FOR MEMORY ERROR PROBS.
    THIS FORMAT IS NOW EXCLUDED SINCE MEMORY IS SEEMINGLY WELL DEBUGED
    SWITCH FORMAT SF:=
    ("LIBRARY MAINTENANCE IN PROGRESS."),
    ("SYSTEM ERROR--MEMORY ACCESS WITH EXPRESSION FOR N OR M."),
    ("SYSTEM ERROR--IMPROPER ARGUMENTS TO FREEPAGE."),
    ("SYSTEM ERROR--TOO LARGE A SUBSCRIPT FOR TYPE SPECIFIED."),
    ("SYSTEM ERROR--TYPE CANNOT BE ZERO WHEN INSERTING OR DELETING."),
    ("SYSTEM ERROR--CHARACTER STRING TOO LONG TO STORE."),
    ("SYSTEM ERROR--ATTEMPT TO INSERT NON-SEQUENTIAL ELEMENT",
     "IN TYPE A STORAGE."),
    ("SYSTEM ERROR--NO BLANKS IN PAGES."),
    ("SYSTEM ERROR--ATTEMPTED BINARY SEARCH OF UNORDERED DATA."),
    ("SYSTEM ERROR--BINARY SEARCH OF UNALLOCATED DATA ATTEMPTED."),
    ("SYSTEM ERROR--BINARY SEARCH FOUND A BLANK PAGE."),
    ("SYSTEM ERROR--DELETION OF TYPE B STORAGE NOT IMPLEMENTED."),
    ("SYSTEM ERROR--ATTEMPT TO DELETE FROM NON-EXISTENT STORAGE."),
    ("SYSTEM ERROR--ATTEMPT TO DELETE RECORD FROM OUTSIDE",
     " ALLOCATED STORAGE."),
    ("SYSTEM ERROR--ATTEMPTED MEMORY SEARCH WITH -N- TOO LARGE."),
    ("SYSTEM ERROR--ATTEMPT TO CHANGE PREVIOUSLY DESIGNATED STORAGE",
     " KIND"),
    ("SYSTEM ERROR--POINTERS TO DATA TYPES OVERLAP."),
    (" ");
    WRITE(PRINT,F,I);
    IF I GTR 0 THEN
        BEGIN
        INTEGER GT1,GT2,GT3;
        MEMORY(10,GT1,GTA,GT2,GT3);
        GO TO FINIS;
        END;
    END;
PROCEDURE MEMORY(MODE,TYPE,A,N,M); VALUE MODE,TYPE;
    INTEGER MODE,TYPE,N,M; ARRAY A[0];
    BEGIN
DEFINE T64= DI:=LOC T; DI:=DI+1; DS:=7 CHR#;
STREAM PROCEDURE WRITERECS(PAGE,A,SKP,NB,NR,NS,RL);
    VALUE SKP,NB,NR,NS,RL;
    BEGIN
    COMMENT -- NS IS THE NUMBER OF WORDS TO SAVE (ON THE
    TAIL OF THE PAGE);
    LOCAL T,T1,T2,TT;
    COMMENT -- MOVE TO POSITION FOR WRITE;
    SI:=LOC NB; T64; SI:=PAGE; SKP(SI:=SI+8);
    T(2(32(RL(SI:=SI+8)))); NB(RL(SI:=SI+8));
    T1:=SI; COMMENT -- RECORDS WILL BE WRITTEN HERE;
    COMMENT -- SKIP OVER TO END OF RECORDS TO BE SAVED;
    DI:=LOC TT; SI:=LOC NS; DI:=DI+1; DS:=7CHR;
    SI:=T1; COMMENT MOVE TO THE END OF THE FIELD TO BE SAVED;
    TT(2(32(RL(SI:=SI+8)))); NS(RL(SI:=SI+8));
    T2:=SI; COMMENT -- END OF FIELD TO BE SAVED;
    SI:=LOC NR; T64; DI:=T2;
    T(2(32(RL(DI:=DI+8)))); NR(RL(DI:=DI+8));
    SI:=T2; SI:=SI-8; DI:=DI-8;
    TT(2(32(RL(DS:=WDS; SI:=SI-16; DI:=DI-16))));
    NS(RL(DS:=WDS; SI:=SI-16; DI:=DI-16));
    COMMENT -- HAVE ACCOMPLISHED THE "SAVE", NOW DO THE WRITE;
    SI:=A; DI:=T1;
    T(2(32(DS:=RL WDS))); NR(DS:=RL WDS)
    END;
STREAM PROCEDURE READRECS(PAGE,A,SKP,NB,NR,NM,RL);
    VALUE SKP,NB,NR,NM,RL;
    BEGIN
    COMMENT
        SKP  = "SKIP"    - - THE NUMBER OF WORDS TO JUMP OVER
        NB   = "NUMBER BEFORE" -- "     " RECORDS TO SKIP BEFORE
               READING THE RECORD.
        NR   = "NUMBER OF RECORDS"  " "  "     "  READ FROM THE
               BUFFER.
        NM   ="NUMBER TO MOVE"   - - "   "  "    "  MOVE OVER TO
               THE PREVIOUSLY READ AREA.
        RL   ="RECORD LENGTH"  -  - THE LENGTH OF EACH ITEM
        ;
    LOCAL T,T1,T2;
    SI:=LOC NB; T64; SI:=PAGE; SKP(SI:=SI+8);
    T(2(32(RL(SI:=SI+8)))); NB(RL(SI:=SI+8));
    T1:=SI;
    COMMENT - - T1 NOW HAS THE STARTING POSITION FOR THE READ;
    SI:=LOC NR; T64; SI:=T1; DI:=A;
    T(2(32(DS:=RL WDS))); NR(DS:=RL WDS);
    T2:=SI; COMMENT T2 CONTAINS THE END OF THE READ;
    SI:=LOC NM; T64; SI:=T2; DI:=T1;
    T(2(32(DS:=RL WDS))); NM(DS:=RL WDS)
    END READRECS;
DEFINE MOVEALONG=
    DI:=LOC C; DI:=DI+6; DS:=2CHR; DI:=LOC Z;
    TSI:=SI; TALLY:=TALLY+1;
    IF TOGGLE THEN
        BEGIN SI:=LOC C; SI:=SI+6;
        IF 2 SC NEQ DC THEN
            BEGIN TAL:=TALLY; SI:=LOC TAL; SI:=SI+7;
            IF SC="0" THEN
                BEGIN TALLY:=TMP; TALLY:=TALLY+1; TMP:=TALLY;
                TALLY:=0;
                END ELSE
                BEGIN SI:=LOC Z; IF SC LEQ"9" THEN ;
                END
            END ELSE
            BEGIN  DI:=TDI; SI:=LOC SIZE; SI:=SI+6; DS:=2CHR;
            TDI:=DI; SI:=SI-2; DI:=LOC C64; DI:=DI+7 ; DS:=CHR;
            SI:=NEW; DI:=TDI; C64(2(DS:=32CHR)); DS:=SIZE CHR;
            TDI:=DI; SI:=TSI; DI:=LOC C; DI:=DI+6;
            DS:=2CHR; TSI:=SI;
            TALLY:=TAL;CHRSTORE:=TALLY; SI:=LOC TMP; SI:=SI+7;
            DI:=LOC CHRSTORE; DI:=DI+6; DS:=CHR END
        END;
    SI:=LOC C; DI:=LOC C64; DI:=DI+1; DS:=7CHR; DI:=TDI; SI:=SI-1;
    DS:=2CHR; SI:=TSI;
    C64(2(DS:=32CHR)); DS:=C CHR; TDI:=DI; TSI:=SI#;
INTEGER STREAM PROCEDURE CHRSTORE(A,SKP,B,NEW,NB,SIZE,NA,MODE,
    PAGESIZE); VALUE SKP,NB,SIZE,NA,MODE,PAGESIZE;
    BEGIN LOCAL T,C,TSI,TDI,
    Z,C64,TMP,TAL;
    LABEL DONE;
    SI:=LOC NB; T64;
    SI:=LOC MODE; SI:=SI+7;
    IF SC="0" THEN ; COMMENT SET TOGGLE;
    SI:=A; DI:=B; SKP(DS:=8CHR);
    TSI:=SI; TDI:=DI;
    T(2(32(MOVEALONG)));  NB(MOVEALONG);
    COMMENT NOW HAVE MOVED UP TO NB;
    IF TOGGLE THEN
        BEGIN TALLY:=TAL; CHRSTORE:=TALLY; SI:=LOC TMP; SI:=SI+7;
        DI:=LOC CHRSTORE; DI:=DI+6; DS:=CHR;
        SI:=LOC SIZE; SI:=SI+6; DI:=TDI; DS:=2CHR; TDI:=DI;
        SI:=LOC SIZE; DI:=LOC C64; DI:=DI+1; DS:=7CHR; SI:=NEW;
        DI:=TDI; C64(2(DS:=32CHR)); DS:=SIZE CHR;
        END ELSE
        BEGIN TSI:=SI; TDI:=DI;
        SI:=LOC MODE; SI:=SI+7;
        IF SC="1" THEN
        COMMENT REMOVE AN ENTRY HERE;
            BEGIN DI:=LOC C; DI:=DI+6; SI:=TSI; DS:=2CHR;
            TSI:=SI; DI:=LOC C64; DI:=DI+1; SI:=LOC C;
            DS:=7CHR; SI:=TSI;  C64(2(SI:=SI+32)); SI:=SI+C;
            TSI:=SI; DI:=LOC CHRSTORE; SI:=LOC C; DS:=WDS;
            DI:=TDI; DS:=2LIT"0"; TDI:=DI;
            END ELSE
        IF SC="2" THEN
        COMMENT READ OUT AN ENTRY;
            BEGIN DI:=LOC C; DI:=DI+6; SI:=TSI; DS:=2CHR;
            TSI:=SI; DI:=LOC C64; DI:=DI+1; SI:=LOC C;
            DS:=7CHR; SI:=TSI; DI:=NEW;
            C64(2(DS:=32CHR)); DS:=C CHR;
            SI:=LOC C; DI:=LOC CHRSTORE; DS:=WDS; GO DONE END;
        SI:=LOC NA; T64; SI:=TSI; DI:=TDI;
        T(2(32(TDI:=DI; DI:=LOC C; DI:=DI+6; DS:=2CHR;
        TSI:=SI; SI:=LOC C; DI:=LOC C64; DI:=DI+1; DS:=7CHR;
    SI:=SI-1;DI:=TDI;DS:=2CHR;SI:=TSI;C64(2(DS:=32CHR));DS:=C CHR)));
        NA( TDI:=DI; DI:=LOC C; DI:=DI+6; DS:=2CHR; TSI:=SI;
        SI:=LOC C;DI:=LOC C64;DI:=DI+1;DS:=7CHR;SI:=SI-1;
       .DI:=TDI;DS:=2CHR;SI:=TSI;C64(2(DS:=32CHR));DS:=C CHR);
        END;
    SI:=LOC PAGESIZE; T64; SI:=B; DI:=A;
%CARD LIST UNSAFE
COMMENT $CARD LIST UNSAFE;
    T(2(DS:=32WDS)); DS:=PAGESIZE WDS;
%CARD LIST SAFE
COMMENT $CARD LIST SAFE;
    DONE:
    END;
STREAM PROCEDURE SETNTH(P,K,N); VALUE K,N;
    BEGIN DI:=P; SI:=LOC K; N(DI:=DI+8); DS:=WDS END;
BOOLEAN STREAM PROCEDURE LESS(A,AN,B,BN,K); VALUE K,AN,BN;
    BEGIN
    SI:=A; DI:=B; SI:=SI+AN; DI:=DI+BN;
    IF K SC LSS DC THEN TALLY:=1;
    LESS:=TALLY
    END;
REAL STREAM PROCEDURE ADDD(A,B); VALUE A,B;
    BEGIN SI:=LOC A; DI:=LOC B; DS:=8ADD; SI:=LOC B;
    DI:=LOC ADDD; DS:=WDS
    END;
INTEGER PROCEDURE FREEPAGE(INDEX,TYPEZERO,START,FINISH);
    VALUE TYPEZERO,START,FINISH; INTEGER TYPEZERO,START,FINISH;
    ARRAY INDEX[0,0];
    IF START GTR FINISH THEN MESSAGE(2) ELSE
    BEGIN ARRAY T[0:RECSIZE+EXTRAROOM+SKIP-1],P[0:FINISH-START];
    INTEGER I,J,K,R;
    R:=RECSIZE+EXTRAROOM+SKIP;
    J:=START-(FINISH+1);
    FOR I:=FINISH STEP -1 UNTIL TYPEZERO DO
```

```
      IF K:=(I+J) LSS TYPEZERO THEN
         BEGIN T[R-1]:=P[TYPEZERO-K-1];
         MOVE(T,R,INDEX[I,0])
         END ELSE
         BEGIN IF I GEQ START THEN P[FINISH-I]:=INDEX[I,R-1];
         MOVE(INDEX[K,0],R,INDEX[I,0]);
         END;
   FREEPAGE:=TYPEZERO-J;
   END;
INTEGER PROCEDURE SEARCHL(A,B,N,MIN,MAX,NP); VALUE N,MIN,MAX;
   INTEGER N,MIN,MAX,NP;
   ARRAY A[0,0]; REAL B;
   BEGIN
   INTEGER I,T;
   FOR I:=MIN STEP 1 WHILE T:=T+A[I,0].CF LEQ B AND I LSS MAX-1 DO;
   IF T LSS B THEN
      BEGIN MESSAGE(3); SEARCHL:=NP:=0;
      END ELSE
      BEGIN SEARCHL:=I; NP:=B-T+A[I,0].CF
      END
   END;
PROCEDURE SORT(A,P,N,C); VALUE P,N,C; INTEGER P,N,C;
   ARRAY A[0,0];
   BEGIN INTEGER R;
      BEGIN
      ARRAY T[0:R:=RECSIZE+EXTRAROOM+SKIP-1];
      LABEL ENDJ;
      INTEGER I,J,L,K,M,SK; R:=R+1;
      SK:=SKIP TIMES 8;
      K:=N-P+1; I:=1; DO UNTIL (I:=I TIMES 2) GTR K;
      M:=I-1;
      WHILE (M:=M DIV 2) NEQ 0 DO
         BEGIN K:=N-M; J:=P;
         DO  BEGIN
            L:=(I:=J)+M;
            DO  BEGIN
            IF A[L,0].TF GTR A[I,0].TF THEN GO ENDJ;
            IF A[L,0].TF EQL A[I,0].TF THEN
               IF NOT(LESS(A[L,0],SK,A[I,0],SK,C)) THEN
               GO ENDJ;
               MOVE(A[L,0],R,T); MOVE(A[I,0],R,A[L,0]);
               MOVE(T,R,A[I,0]);
               END UNTIL (I:=(L:=I)-M) LSS P;
            ENDJ:
            END UNTIL (J:=J+1) GTR K;
         END
      END
   END SORT;
   COMMENT - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         MODE                                            MEANING
         ----                                            -------
          1 =                                            INTERROGATE TYPE
          2 =                                     INSERT RECORD REL ADDRS N
                                                  (RELATIVE TO START OF LAST PAGE)
          3 =                                     RETURN THE NUMBER OF RECORDS (M)
          4 =                                        "    ITEM AT RECORD # N
          5 =                                     INSERT  "    "     "    " "
          6 =                                     DELETE  "    "     "    " "
          7 =                                     SEARCH FOR THE RECORD -A-
          8 =                                     FILE OVERFLOW, INCREASE BY N
          9 =                                     FILE MAINTENANCE
         10 =                                     EMERGENCY FILE MAINTENANCE
          11                                      SET STORAGE KIND
        12=                                      ALTER STORAGE ALLOCATION RESOURCES
        13=                                      RELEASE "TYPE" STORAGE TO SYSTEM
        14=                                      CLOSE ALL PAGES FOR AREA TRANSITION
   NOTE THAT WHEN SEQUENTIAL STORAGE MAINTENANCE IS DONE, N
   WILL ALWAYS INDICATE THE ADDRESS OF THE STRING RELATIVE TO
   THE TYPE SPECIFIED, AND M WILL ALWAYS BE THE LENGTH OF THE
   STRING IN -A- (EITHER AS INPUT OR OUTPUT)
   ;
   PROCEDURE UPDATE(T,L,U,D); VALUE L,U,D; INTEGER L,U,D;
      ARRAY T[0];
      BEGIN INTEGER I,J,K;
      FOR I:=L STEP 1 UNTIL U DO
         BEGIN J:=T[I].AF+D; T[I].AF:=J;
         J:=T[I].BF+D; T[I].BF:=J
         END
      END;
   OWN INTEGER CURPAGE,NPAGES,NTYPES,P,PS,U,L;
   OWN INTEGER FIRST,AVAIL,MAXBUFF,CURBUFF;
REAL GT1;
LABEL MOREPAGES;
```

```
IF MEMBUG.[21:1] THEN DUMPMEMORY(MODE,TYPE,N,M);
IF MODE=8 THEN NPAGES:=NPAGES+N;
MOREPAGES:
   BEGIN
   OWN BOOLEAN POINTERSET, TYPESET;
   INTEGER I, T, NR;
   OWN ARRAY BUF[0:MBUFF], TYPS[0:511];
      OWN ARRAY INDX[0:NPAGES,0:RECSIZE+EXTRAROOM+SKIP-1];
   PROCEDURE SETTYPES;
   BEGIN INTEGER I, T;
       FOR I := 0 STEP 1 UNTIL NPAGES DO
       IF INDX[I,0].TF NEQ T THEN
       BEGIN
           TYPS[T].BF := I; TYPS[T:=INDX[I,0].TF].AF := I;
           TYPS[T].BOOL := INDX[I,0].MF;
       END;
       TYPS[T].BF := I;
   END SETTYPES;
   REAL PROCEDURE BUFFNUMBER(I); VALUE I; INTEGER I;
      BEGIN INTEGER K,L,M;
      LABEL D;
      DEFINE B=BUF#;
      IF( IF K:=INDX[I,P].BUFF=0 THEN TRUE ELSE BUF[K].PAGEF
         NEQ INDX[I,P].PAGEF+1) THEN
         BEGIN IF NULL(K:=CDR(AVAIL)) THEN
            BEGIN K:=CDR(FIRST);
            WHILE M:=CDR(B[K]) NEQ 0 DO
               BEGIN L:=K; K:=M; END;
            RPLACD(B[L],0);
            IF BOOLEAN(B[K].CHANGEDBIT) THEN
               WRITE(POINTERS[K][B[K].PAGEF-1]);
            B[K].CHANGEDBIT:=0;
            END ELSE  RPLACD(AVAIL,CDR(B[K]));
         B[K].PAGEF:=INDX[I,P].PAGEF+1;
         INDX[I,P].BUFF:=K;
         READ(POINTERS[K][INDX[I,P].PAGEF]);
         END ELSE
         IF CDR(FIRST)=K THEN GO TO D ELSE
            BEGIN L:=CDR(FIRST);
            WHILE M:=CDR(B[L]) NEQ K DO L:=M;
            RPLACD(B[L],CDR(B[M]));
            END;
      RPLACD(B[K],CDR(FIRST)); RPLACD(FIRST,K);
      D: BUFFNUMBER:=K
      END;
   PROCEDURE MARK(I); VALUE I; INTEGER I;
         BUF[INDX[I,P].BUFF].CHANGEDBIT:=1;
BOOLEAN PROCEDURE WRITEBUFFER;
   BEGIN INTEGER I;
   I:=CDR(FIRST);
   WHILE NOT NULL(I) DO
      IF BOOLEAN(BUF[I].CHANGEDBIT) THEN
      BEGIN WRITEBUFFER:=TRUE;
      BUF[I].CHANGEDBIT:=0;
      WRITE(POINTERS[I][BUF[I].PAGEF-1]);
      RPLACD(I,0);
      END ELSE I:=CDR(BUF[I]);
   END;
   IF NOT POINTERSET THEN
      BEGIN LABEL EOF;
      READ(POINTERS[1][NPAGES])[EOF];
      IF EOFMARK(SKIP,RECSIZE,POINTERS[1](0))THEN GO TO EOF;
      MOVE(POINTERS[1](0),1,T);
         COMMENT -- USE T TO DETERMIN THE VARIABLE REC SIZE LATER;
      MOVE(POINTERS[1](0),RECSIZE+SKIP,INDX[NPAGES,0]);
         INDX[NPAGES,RECSIZE+1].PAGEF:=NPAGES;
         NPAGES:=NPAGES+1;
      GO TO MOREPAGES;
      COMMENT - - INTIALIZE VARIABLES;
   EOF: POINTERSET:=TRUE;
   U:=PAGESIZE-SKIP-PAGESPACE;
   L:=(U-ALLOWANCE)/RECSIZE;
   U:=(U+ALLOWANCE+RECSIZE/2)/RECSIZE;
   PS:=(U+L)/2;
   CURPAGE:=NPAGES:=NPAGES-1;
   CURBUFF:=1;
   P:=RECSIZE+SKIP;
   FOR T:=1 STEP 1 UNTIL SBUFF DO RPLACD(BUF[T],T+1);
   RPLACD(BUF[SBUFF],0); RPLACD(AVAIL,1);
   MAXBUFF:=SBUFF;
         T:=0;
   SORT(INDX,0,NPAGES,RECSIZE TIMES 8);
```

```
    FOR I:=0 STEP 1 UNTIL NPAGES DO
        IF INDX[I,0].TF GTR T THEN T:=INDX[I,0].TF;
    NTYPES:=T;
    END;
    IF TYPE GTR NTYPES THEN NTYPES:=TYPE;
        IF NOT TYPESET THEN
        BEGIN TYPESET:=TRUE; SETTYPES;
        COMMENT
        IF MEMBUG THEN DUMPINDEX(TYPS,NTYPES,INDX,RECSIZE,
        P);
        END;
COMMENT --- DECIDE WHETHER TO SAVE CURRENT PAGE BEFORE GOING ON;
IF MODE=2 THEN
    BEGIN  MODE:=5; NR:=N
    END ELSE
IF MODE GEQ 4 THEN                           %MAY BE FILE MAINTENANCE
    IF MODE GEQ 8 THEN                       %IS FILE MAINTENANCE
    ELSE                                     %WE MAY BE GOING TO
    IF MODE NEQ 7 THEN                       %ANOTHER PAGE
        BEGIN
        IF TYPE=0 THEN BEGIN MESSAGE(4); MODE:=0 END ELSE
        IF TYPS[TYPE].AF=TYPS[TYPE].BF THEN
            IF TYPS[0].BF GTR 0 THEN
            BEGIN INTEGER J,K; REAL PG;
            K:=TYPS[0].BF-1; TYPS[0].BF:=K; PG:=INDX[K,P];
            FOR I:=1 STEP 1 UNTIL TYPE-1 DO
            IF (T:=TYPS[I]).AF NEQ T.BF THEN
                BEGIN FOR K:=T.AF STEP 1 UNTIL T.BF -1 DO
                MOVE(INDX[K,0],P+EXTRAROOM,INDX[K-1,0]);
                TYPS[I].AF:=T.AF-1; TYPS[I].BF:=K:=T.BF-1
                END;
            IF CURPAGE GTR TYPS[0].BF THEN
                IF CURPAGE LEQ K THEN CURPAGE:=CURPAGE-1;
            TYPS[TYPE].BF:=K+1; TYPS[TYPE].AF:=K;
            INDX[K,P]:=PG; INDX[K,0]:=0; INDX[K,0].TF:=TYPE;
            IF TYPS[TYPE].BOOL=1 THEN
                BEGIN SETNTH(INDX[K,0],0,1); INDX[K,0].MF:=1
                END;
            COMMENT
            IF MEMBUG.[22:1] THEN DUMPTYPES(MODE,TYPS,NTYPES);
            MEMORY(MODE,TYPE,A,N,M);  MODE:=0
            END ELSE
            BEGIN T:=1; MEMORY(8,TYPE,A,T,M); MEMORY(MODE,TYPE,A,N,M);
            MODE:=0
            END ELSE
    IF NOT( BOOLEAN(TYPS[TYPE].BOOL) AND MODE=5) THEN
        CURBUFF:=BUFFNUMBER(CURPAGE:=
            SEARCHL(INDX,N,NPAGES,TYPS[TYPE].AF,TYPS[TYPE].BF,
                NR)  ));
    COMMENT
      IF MEMBUG.[23:1] THEN CHECKTYPES(TYPS,NTYPES);
        END;
    COMMENT
    IF MEMBUG.[20:1] THEN DUMPINDEX(TYPS,NTYPES,INDX,RECSIZE,P);
    COMMENT
    IF MEMBUG.[24:1] THEN DUMPBUFF(BUF,FIRST,AVAIL);
CASE MODE OF
    BEGIN
    %------- MODE=0 ------- RESERVED ---------------
    ;
    %------- MODE=1 ------------------------------------------------------
    IF M=0 THEN N:=TYPS[TYPE].BOOL ELSE
    IF M=1 THEN
        BEGIN FOR I:=1 STEP 1 UNTIL NTYPES DO
        IF (T:=TYPS[I]).AF=T.BF THEN
            BEGIN N:=I; I:=NTYPES+1
            END;
        IF I=NTYPES+1 THEN N:=NTYPES+1
        END;
    %------- MODE=2 ------- RESERVED ---------------
    ;
    %------- MODE=3 ------- RETURN THE NUMBER OF RECORDS----
    BEGIN COMMENT IF TYPE LSS 0 THEN THE TOTAL NUMBER
    OF PAGES IS GIVEN, OTHERWISE THE NUMBER OF "TYPE" PAGES IS
    GIVEN;
    FOR I:=0 STEP 1 UNTIL NPAGES DO
        IF INDX[I,0].TF=TYPE OR TYPE LSS 0 THEN
        NR:=NR+INDX[I,0].CF;
    M:=NR
    END;
    %------- MODE=4 ------- RETURN ITEM AT SUBSCRIPT N -----
    IF NR GEQ INDX[CURPAGE,0].CF THEN MESSAGE(3) ELSE
    IF BOOLEAN(TYPS[TYPE].BOOL) THEN COMMENT SEQUENTIAL STORAGE;
```

```
          BEGIN ARRAY B[0:PAGESIZE];
          M:=CHRSTORE(POINTERS[CURBUFF](0),2,B,A,NR,0,0,2,0);
          END ELSE
     BEGIN
     M:=RECSIZE×8;
     READRECS(POINTERS[CURBUFF](0),A,SKIP,NR,1,0,RECSIZE);
     END;
     %------- MODE=5 ------- INSERT ITEM AT SUBSCRIPT N;
     BEGIN INTEGER K,J,S; REAL PG;
     IF BOOLEAN(TYPS[TYPE].BOOL) THEN
     COMMENT FIND A PLACE FOR THE CHARACTER STRING OF LENGTH
     M;
     IF M GTR (PAGESIZE-SKIP-1)×8-2 THEN MESSAGE(5) COMMENT
     THIS CHARACTER STRING IS TOO LONG ; ELSE
          BEGIN ARRAY C[0:PAGESIZE];
          STREAM PROCEDURE ADDZERO(CHARS,POINTER); VALUE CHARS;
               BEGIN LOCAL T;
               SI:=LOC CHARS; DI:=LOC 1; DI:=DI+1; DS:=7CHR;
               DI:=POINTER; T(2(DI:=DI+32)); CHARS(DI:=DI+1);
               DS:=2LIT"0";
               END;
          BOOLEAN B,NOTLASTPAGE;
          LABEL TRYITAGAIN;
          TRYITAGAIN:
          FOR I:=(T:=TYPS[TYPE]).AF STEP 1 WHILE I LSS T.BF AND
          NOT B DO
               IF NOT(B:=((PAGESIZE-SKIP-1)×8-(GT1:=INDX[I,0]).SF)GEQ M+2
               AND NOT BOOLEAN(GT1.FF)) THEN S:=S+GT1.CF ELSE I:=I-1;
          NOTLASTPAGE:=B AND I NEQ T.BF-1;
          COMMENT IF B IS TRUE, THEN A PAGE HAS BEEN FOUND;
          IF NOT B THEN COMMENT GET A PAGE THAT IS FREE;
               BEGIN
               COMMENT
               IF MEMBUG.[5:1] THEN DUMPTYPES(5,1,TYPS,NTYPES);
                    IF TYPS[0].BF=0 THEN BEGIN K:=CURPAGE; T:=1;
                    MEMORY(8,TYPE,A,T,M); CURPAGE:=K+1
                    END
               ELSE
               IF (PAGESIZE-SKIP-1)×8-INDX[(I:=I-1)-1,0].SF GTR 2 THEN
                    BEGIN
                    CURBUFF:=BUFFNUMBER(CURPAGE:=I-1);
                    ADDZERO((GT1:=INDX[CURPAGE,0].SF)+8×(SKIP+1),POINTERS
                         [CURBUFF](0));
                    INDX[CURPAGE,0].SF:=GT1+2;
                    INDX[CURPAGE,0].CF:=INDX[CURPAGE,0].CF+1;
                    COMMENT SINCE ALLOCATING A NEW PAGE, SET COUNT TO
                    ONE MORE AND FREEZE THE COUNT;
                    S:=S+1; % SINCE THE COUNT INCREASED
                    MOVE(INDX[CURPAGE,0],SKIP+1,POINTERS[CURBUFF](0));
                    MARK(CURPAGE);
                    END;
               T:=TYPS[0].BF; TYPS[0].BF:=T:=T-1;
               COMMENT T IS THE SUBSCRIPT INTO THE NEW PAGE;
               PG:=INDX[T,P]; COMMENT PG HOLDS THE NEW PAGE #;
               FOR K:=T+1 STEP 1 UNTIL I DO
                    MOVE(INDX[K,0],RECSIZE+SKIP+EXTRAROOM,INDX[K-1,0]);
               T:=TYPS[TYPE].AF; TYPS[TYPE].AF:=T-1;
               INDX[I,P]:=PG; UPDATE(TYPS,1,TYPE-1,-1);
               IF CURPAGE GTR TYPS[0].BF THEN IF CURPAGE LEQ
                    I THEN CURPAGE:=CURPAGE-1;
               INDX[I,0]:=0; INDX[I,0].MF:=1; INDX[I,0].TF:=TYPE;
               COMMENT MUST ALSO ASSIGN A NUMBER TO THIS PAGE
               (TO BE STORED IN THE PAGE) TO KEEP IT IN SEQUENCE
               WITHIN THIS TYPE;
               IF (T:=TYPS[TYPE]).AF LSS T.BF-1 THEN
                    T:=INDX[T.BF-1,1] ELSE T:=0;
               SETNTH(INDX[I,0],ADDD(1,T),1);
               COMMENT END OF THE INITIALIZATION OF THE INDEX ARRAY,
               WE STILL HAVE TO MAKE SOME ENTRIES INTO THE PAGE
               WHICH WE WILL DO BELOW;
               END OF TEST FOR NEW PAGE;
          COMMENT I IS SET TO THE PROPER SUBSCRIPT FOR THE CHR STORE;
               CURBUFF:=BUFFNUMBER(CURPAGE:=I);
          COMMENT NOW THE CORRECT PAGE IS IN CORE.
               ---------------------------
               M= NUMBER OF CHARACTERS IN A (ON INPUT)
               N= ADDRESS OF A WITHIN THIS TYPE (ON OUTPUT
               ---------------------------;
          K:=INDX[I,0];
          T:=CHRSTORE(POINTERS[CURBUFF](0),SKIP+1,C,A,K.CF,M,0,0,
               PAGESIZE);
          COMMENT K.CF IS THE NUMBER OF ITEMS ALREADY IN THIS
          PAGE.  IF THERE IS A SEGMENT WHICH IS NULL, IT WILL
```

```
    BE FOUND AND ASSIGNED AS THE SEG NUMBER FOR
    THIS CHARACTER STRING (T).  IF NOT, IT WILL STICK THE
    STRING ON THE END (WE KNOW THERE IS ENOUGH ROOM
    SINCE WE CHECKED INDX[I,0].SF -- THE NUMBER OF CHRS USED
    IN THIS PAGE, OR WE CREATED A NEW PAGE);
    N:=S+T; S:=K.SF; COMMENT S CONTAINS THE # OF CHRS USED UP;
    IF T:=T+1 GTR K.CF THEN COMMENT ADDED THE STRING ON THE END;
        IF NOTLASTPAGE THEN % PAGE ALREADY FULL
            BEGIN S:=0; B:=FALSE; INDX[I,0].FF:=1;
            MOVE(INDX[I,0],SKIP+1,POINTERS[CURBUFF](0));
            MARK(CURPAGE); GO TRYITAGAIN; END ELSE
        BEGIN K.CF:=T; S:=S+2;
        END
    ELSE IF T=K.CF AND NOTLASTPAGE THEN INDX[I,0].FF:=1;

    INDX[I,0].CF:=K.CF; INDX[I,0].SF:=S+M;
    MOVE(INDX[I,0],SKIP+1,POINTERS[CURBUFF](0));
    MARK(CURPAGE);
    COMMENT THE PAGE DESCRIPTOR HAS BEEN UPDATED;
    COMMENT
        IF MEMBUG.[5:1] THEN DUMPTYPES(5,2,TYPS,NTYPES);
    END ELSE COMMENT KIND OF STORAGE IS SORTED;
IF NR GTR (T:=INDX[CURPAGE,0].CF) THEN
COMMENT SUBSCRIPT IS NOT IN THE MIDDLE OF THE PAGE;
MESSAGE(6) ELSE
    BEGIN
    IF T GEQ U THEN COMMENT WILL EXCEED UPPER PAGE BOUND;
        BEGIN ARRAY B[0:RECSIZE TIMES
        (T-PS+(I:=(IF NR GEQ PS THEN 0 ELSE 1)))-1];
        COMMENT B IS JUST BIG ENOUGH TO CARRY THE
        EXCESS FROM THE OLD PAGE;
        READRECS(POINTERS[CURBUFF](0),B,SKIP,PS-I,
            J:=(T-PS+I),0,RECSIZE);
        COMMENT -- B NOW HAS THE EXCESS;
        INDX[CURPAGE,0].CF:=T-J; SETNTH(POINTERS[CURBUFF](0),
        INDX[CURPAGE,0],0);
        MARK(CURPAGE);
        IF TYPS[0].BF=0 THEN
            BEGIN K:=CURPAGE; T:=1;
            MEMORY(8,TYPE,A,T,M); CURPAGE:=K+1;
            END;
        COMMENT -- ASSIGN A FREE PAGE (SUBS T);
        T:=TYPS[0].BF; TYPS[0].BF:=T:=T-1;

        PG:=INDX[T,P];
        FOR K:=T+1 STEP 1 UNTIL CURPAGE DO
        MOVE(INDX[K,0],RECSIZE+SKIP+EXTRAROOM,INDX[K-1,0]);
        INDX[CURPAGE,P]:=PG;
        T:=0;T.CF:=J;T.TF:=TYPE;
        CURBUFF:=BUFFNUMBER(CURPAGE);
        WRITERECS(POINTERS[CURBUFF](0),B,SKIP,0,J,0,RECSIZE);
        SETNTH(POINTERS[CURBUFF](0),T,0);
        MOVE(POINTERS[CURBUFF](0),RECSIZE+SKIP,INDX[CURPAGE,0]);
        MARK(CURPAGE);
        T:=TYPS[TYPE].AF; TYPS[TYPE].AF:=T-1;
        UPDATE(TYPS,1,TYPE-1,-1);
        IF J=0 THEN MESSAGE(7);
        IF BOOLEAN (I) THEN
        COMMENT    I=0 IMPLIES THE RECORD GOES TO NEW PAGE,
                   I=1 IMPLIES THE RECORD GOES TO NOOLD PAGE;
            BEGIN
            T:=INDX[CURPAGE:=CURPAGE-1,0].CF;
            CURBUFF:=BUFFNUMBER(CURPAGE);
            ; COMMENT OLD PAGE IS NOW BACK;
            END ELSE
            BEGIN T:=J; NR:=NR-PS
            END
        END;
    WRITERECS(POINTERS[CURBUFF](0),A,SKIP,NR,1,T-NR,RECSIZE);
    T:=INDX[CURPAGE,0].CF; INDX[CURPAGE,0].CF:=T+1;
    SETNTH(POINTERS[CURBUFF](0),INDX[CURPAGE,0],0);
    IF NR=0 THEN MOVE(POINTERS[CURBUFF](0),RECSIZE+SKIP,INDX
    [CURPAGE,0]);  MARK(CURPAGE);
    END;
END;
%------- MODE=6 ------- DELETE A RECORD FROM THE FILE ----
IF (T:=TYPS[TYPE]).AF=T.BF THEN MESSAGE(12) COMMENT
ATTEMPT TO DELETE NON-EXISTENT STORAGE;
ELSE
IF NR GEQ(T:=INDX[CURPAGE,0].CF) THEN MESSAGE(13) COMMENT
ATTEMPT TO DELETE OUTSIDE STORAGE RANGE; ELSE
IF BOOLEAN(T.BOOL) THEN COMMENT SEQUENTIAL STORAGE;
    BEGIN COMMENT NR IS THE RECORD TO DELETE;
```

```
        ARRAY B[0:PAGESIZE-1];
        COMMENT PAGESIZE -1 SHOULD BE COMPUTED TO THE EXACT
        NUMBER OF WORDS TO MOVE -- IT WOULD SPEED THINGS UP;
        INTEGER L;
        T:=INDX[CURPAGE,0]; COMMENT T.CF IS THE NUMBER OF
        RECORDS ON THIS PAGE, T.SF IS THE NUMBER OF CHRS;
        L:=CHRSTORE(POINTERS[CURBUFF](0),SKIP+1,B,A,NR,0,T.CF
        -NR-1,1,PAGESIZE);
        COMMENT WE WILL BRING BACK THE NUMBER OF CHRS IN M;
        M:=L;
        MARK(CURPAGE);
        COMMENT MAKE CHANGES TO THE CHARACTER COUNT;
        INDX[CURPAGE,0].SF:=T.SF-L;
        INDX[CURPAGE,0].FF:=0; % PAGE IS CERTAINLY NOT FULL NOW
        COMMENT AND  WE ARE DONE WITH THE DELETION;
        MOVE(INDX[CURPAGE,0],SKIP+1,POINTERS[CURBUFF](0));
        END
    ELSE
    BEGIN ARRAY A[0:RECSIZE-1];
    INDX[CURPAGE,0].CF:=I-1;
    SETNTH(POINTERS[CURBUFF](0),INDX[CURPAGE,0],0);
    IF I GTR 1 THEN
        BEGIN
        READRECS(POINTERS[CURBUFF](0),A,SKIP,NR,1,I-NR-1,RECSIZE);
        MARK(CURPAGE);
        IF NR=0 THEN
            MOVE(POINTERS[CURBUFF](0),RECSIZE+SKIP,INDX[CURPAGE,0])
        END ELSE COMMENT FREE THE EMPTY PAGE;
        BEGIN MARK(CURPAGE);
        ;TYPS[0].BF:=FREEPAGE(INDX,TYPS[0].BF,CURPAGE,CURPAGE);
        UPDATE(TYPS,1,TYPE-1,1); TYPS[TYPE].AF:=T.AF+1;
        COMMENT
        IF MEMBUG.[6:1] THEN DUMPTYPES(MODE,TYPS,NTYPES);
        END
    END;
    %------- MODE=7 ------- SEARCH FOR A RECORD FROM THE FILE ---
    IF N GTR 3 THEN MESSAGE(14) ELSE
COMMENT RETURN RECORD CLOSEST (BUT LESS THAN OR EQUAL TO) TO
    THE CONTENTS OF -A-.  A WILL BE REPLACED BY THE RECORD FOUND;
    IF BOOLEAN((I:=TYPS[TYPE]).BOOL) THEN
    MESSAGE(8) COMMENT BINARY SEARCH OF NON-SEQUENTIAL DATA;
    ELSE
    IF I.AF=I.BF THEN MESSAGE(9) COMMENT --NO STORAGE OF
    THIS TYPE ALLOCATED AS YET;
    ELSE BEGIN
        INTEGER F,U,L;
        ARRAY B[0:RECSIZE-1];
        U:=TYPS[TYPE].BF; L:=TYPS[TYPE].AF;
        WHILE U-L GTR 1 DO
        IF LESS(A,0,INDX[F:=(U+L) DIV 2,0],8,M) THEN U:=F ELSE L:=F;
        CURBUFF:=BUFFNUMBER(CURPAGE:=L);
        L:=0; U:=INDX[CURPAGE,0].CF;
        IF L-U=0 THEN MESSAGE(10) COMMENT BINARY SEARCH FOUND
        A PAGE WITH NO RECORDS;
        ELSE BEGIN
            WHILE U-L GTR 1 DO
            BEGIN READRECS(POINTERS[CURBUFF](0),B,SKIP,
            F:=(U+L) DIV 2,1,0,RECSIZE);
            IF LESS(A,0,B,0,M) THEN U:=F ELSE L:=F
            END;
            COMMENT -----------------------------------
            ON INPUT:
               N=0     IMPLIES    DO NOT PLACE RECORD INTO FILE
                                  IF RECORD IS FOUND. RETURN RELA-
                                  TIVE POSITION OF THE CLOSEST RECORD
                                  IN THIS PAGE.
               N=1        "       DO NOT PLACE IN FILE. RETURN ABSO-
                                  LUTE SUBSCRIPT OF CLOSSEST RECORD.
               N=2        "       PLACE RECORD INTO FILE IF NOT FOUND.
                                  RETURN RELATIVE POSITION OF RECORD.
               N=3        "       PLACE RECORD INTO FILE, IF NOT
                                  FOUND, RETURN ABS SUBSCRIPT OF
                                  THE RECORD.
         ON OUTPUT:
               M=0        "       RECORD FOUND WAS EQUAL TO RECORD
                                  SOUGHT.
               M=1        "       RECORD FOUND WAS GREATER THAN THE
                                  SOUGHT.
               M=2        "       RECORD FOUND WAS LESS THAN THE
                                  RECORD SOUGHT.

            READRECS(POINTERS[CURBUFF](0),B,SKIP,L,1,0,RECSIZE);
            IF LESS(A,0,B,0,M) THEN M:=1 ELSE
            IF LESS(B,0,A,0,M) THEN M:=2 ELSE
            M:=0;
        T:=0; IF BOOLEAN(N) THEN
            FOR I:=TYPS[TYPE].AF STEP 1 UNTIL CURPAGE-1 DO
            T:=T+INDX[I,0].CF;
        IF N GTR 1 THEN IF M GEQ 1 THEN
            MEMORY(2,TYPE,A,L+M-1,NR);
        MOVE(B,RECSIZE,A);
        N:=T+L;
        END
    END;
%------- MODE=8 ------- FILE OVERFLOW, FIX ARRAYS AND PAGES
    BEGIN BOOLEAN TOG;
        ARRAY A[0:PAGESIZE-1]; T:=NPAGES-N+1;
    IF TOG:=(T DIV AREASIZE) LSS (NPAGES DIV AREASIZE ) OR
        (T=NPAGES AND T MOD AREASIZE =0) THEN
        MEMORY(14,TYPE,A,N,M);
    FOR I:=T STEP 1 UNTIL NPAGES DO
    BEGIN WRITE(NEWDISK[I],PAGESIZE,A[*]);INDX[I,P].PAGEF:=I END;
    MARKEOF(SKIP,RECSIZE,NEWDISK(0));
    WRITE(NEWDISK[I]);
    TYPS[0].BF:=FREEPAGE(INDX,TYPS[0].BF,T,NPAGES);
    UPDATE(TYPS,1,NTYPES,NPAGES-T+1);
    IF TOG THEN CLOSE(NEWDISK);
    END;
%------- MODE=9 ------- FILE MAINTENANCE -----------------
BEGIN BOOLEAN ITHPAGEIN;
INTEGER I,J,K,T1,T2,T3,M,W,Q;
ARRAY A,B[0:PAGESIZE-1];
COMMENT
MONITOR PRINT(Q,W,N, I,J,K,T1,T2,T3,M,A,B);
IF I:=TYPS[0].BF LEQ NPAGES THEN
DO
    BEGIN COMMENT OUTER "DO-LOOP" TO FIND TROUBLE WITH
    THE FILE;
    IF T1:=(Q:=INDX[I,0]).CF LSS L THEN COMMENT MAY BE CORRECTABLE;
        IF NOT BOOLEAN((Q:=TYPS[Q.TF]).BOOL) THEN
        COMMENT -- THIS PAGE IS CORRECTABLE;
            IF I NEQ NPAGES THEN
            COMMENT -- THIS IS NOT THE LAST PAGE OF THE FILE;
                IF (J:=I+1) LSS Q.BF THEN
                COMMENT -- THIS IS NOT THE LAST PAGE OF THIS TYPE;
                BEGIN COMMENT -- FIND RECORDS TO MOVE INTO
                THIS PAGE;
                DO IF T2:=INDX[J,0].CF GTR 0 THEN
                COMMENT THIS PAGE HAS RECS TO MOVE;
                    BEGIN COMMENT HOW MANY;
                    IF T2 LSS K:=PS-T1 THEN K:=T2;
                    IF NOT ITHPAGEIN THEN
                        BEGIN COMMENT BRING IN PAGE I;
                        MOVE(POINTERS[BUFFNUMBER(I)](0),
                        PAGESIZE,B); ITHPAGEIN:=TRUE
                        END;
                    COMMENT -- BRING IN PAGE J;
                    CURBUFF:=BUFFNUMBER(CURPAGE:=J);
                    COMMENT -- MOVE SOME INTO A;
                    READRECS(POINTERS[CURBUFF](0),A,SKIP,0,K,
                    T2:=T2-K,RECSIZE); INDX[J,0].CF:=T2;
                    IF T2=0 THEN
                        COMMENT SET THIS PAGE FREE;
                        INDX[J,0]:=0;
                    SETNTH(POINTERS[CURBUFF](0),INDX[J,0],0);
                    MOVE(POINTERS[CURBUFF](0),RECSIZE+SKIP,INDX[J
                    ,0]); MARK(CURPAGE);
                    COMMENT -- PUT THE RECORDS INTO PAGE I;
                    WRITERECS(B,A,SKIP,T1,K,0,RECSIZE);
                    END
                   ELSE K:=0 COMMENT SINCE NO CONTRI-
                   BUTION;
                UNTIL T1:=T1+K GEQ PS OR J:=J+1 GEQ Q.BF;
                INDX[I,0].CF:=T1; B[0]:=INDX[I,0];
                COMMENT -- PUT THE PAGE BACK OUT ON DISK;
                MOVE(B,RECSIZE+SKIP,INDX[I,0]);
                MOVE(B,PAGESIZE,POINTERS[CURBUFF:=BUFFNUMBER
                (I)](0)); SORT(INDX,0,NPAGES,RECSIZE×8);
                MARK(CURPAGE:=I); SETTYPES;
                N:=1;
                END
                ELSE N:=0 COMMENT LAST PAGE OF THIS TYPE;
            ELSE N:=0 COMMENT LAST PAGE OF FILE;
        ELSE N:=0 COMMENT PAGE CANNOT BE CHANGED;
    ELSE N:=0 COMMENT THIS PAGE IS  NOT TOO SMALL;
    END UNTIL I:=I+1 GTR NPAGES OR N NEQ 0 ELSE N:=0;
    IF I GTR NPAGES THEN N:=REAL(WRITEBUFFER);
    END OF FILE UPDATE;
    %------- MODE=10 ------ EMERGENCY FILE MAINTENANCE --------
    DO MEMORY(9,TYPE,A,N,M) UNTIL N NEQ 1
    %------- MODE=11 ------- SET THE KIND OF STORAGE FOR TYPE ----------
   ;COMMENT  TYPE "TYPE" STORAGE IS BEING SET TO SEQUENTIAL;
    IF TYPE=0 THEN MESSAGE(4) ELSE
    IF (T:=TYPS[TYPE]).AF= T.BF THEN TYPS[TYPE].BOOL:=1 ELSE
    MESSAGE(15); COMMENT ATTEMPT TO CHANGE KINDS IN MIDSTREAM;
%------ MODE=12 ----------- ALTER STORAGE ALLOCATION RESOURCES---
    COMMENT N IS THE "FACTOR" (PERCENT OF RESOURCES x 100),
    AND M IS THE STORAGE "LEVEL" (0 IS THE ONLY ONE THAT
    DOES ANYTHING ON THE B5500);
    BEGIN INTEGER J,K;
    BOOLEAN TOG;
    IF T:=Nx(MBUFF-1)/100+1 GTR MAXBUFF THEN
        BEGIN COMMENT ADD TO AVAILABLE LIST;
        FOR I:=CDR(FIRST),CDR(AVAIL) DO
        WHILE NOT NULL(I) DO
            BEGIN BUF[I].FLAGB:=1; I:=CDR(BUF[I]);
            END;
        FOR I:=MAXBUFF+1 STEP 1 UNTIL T DO
            BEGIN WHILE BUF[K:=K+1].FLAGB=1 DO;
            BUF[K]:=0; RPLACD(BUF[K],CDR(AVAIL));
            RPLACD(AVAIL,K)
            END;
        MAXBUFF:=T;
        FOR I:=1 STEP 1 UNTIL MBUFF DO BUF[I].FLAGB:=0;
        END ELSE
    IF T LSS MAXBUFF THEN
        BEGIN COMMENT CUT DOWN ON THE NUMBER OF BUFFERS;
        I:=CDR(FIRST);
        FOR J:=1 STEP 1 UNTIL MAXBUFF DO
            IF TOG THEN
                IF NOT NULL(I) THEN
                    IF J GEQ T THEN
                        BEGIN K:=CDR(BUF[I]); BUF[I]:=0
                        ; I:=K END
                    ELSE I:=CDR(BUF[I])
                ELSE
            ELSE
            IF TOG:=NULL(I) THEN
                BEGIN J:=J-1; I:=CDR(AVAIL)
                END
            ELSE
                IF J EQL T THEN
                    BEGIN K:=CDR(BUF[I]); RPLACD(BUF[I],0);
                    I:=K END ELSE
                IF J GTR T THEN
                    BEGIN
                    IF BOOLEAN(BUF[I].CHANGEDBIT) THEN
                        WRITE(POINTERS[I][BUF[I].PAGEF-1]);
                    K:=CDR(BUF[I]);
                    CLOSE(POINTERS[I]);
                    BUF[I]:=0; I:=K
                    END ELSE I:=CDR(BUF[I])
                ;
        MAXBUFF:=T
        END;
   END;
   %------- MODE=13 ------- RELEASE "TYPE" STORAGE TO SYSTEM ----------
   IF (T:=TYPS[TYPE]).BF GTR T.AF THEN
       BEGIN INTEGER J;
       J:=T.BF-1;
       FOR I:=T.AF STEP 1 UNTIL J DO
           BEGIN CURBUFF:=BUFFNUMBER(I);
           SETNTH(POINTERS[CURBUFF](0),0,0); MARK(CURPAGE:=I);
           END;
       TYPS[0].BF:=FREEPAGE(INDX,TYPS[0].BF,T.AF,J);
       UPDATE(TYPS,1,TYPE-1,J-T.AF+1);
       TYPS[TYPE].BF:=T.AF;  TYPS[TYPE].BOOL:=0;
       END;
   %------- MODE=14 ------ RELEASE ALL PAGES FOR TRANSITION ----------
   BEGIN INTEGER K;
   I:=CDR(FIRST);
   WHILE NOT NULL(I) DO
       BEGIN IF BOOLEAN(BUF[I].CHANGEDBIT ) THEN WRITE(POINTERS[I]
           [BUF[I].PAGEF-1]);  CLOSE(POINTERS[I]);
       K:=CDR(BUF[I]);  BUF[I]:=0;
       RPLACD(BUF[I],CDR(AVAIL)); RPLACD(AVAIL,I); I:=K
       END   ; CURPAGE:=CURBUFF:=-1; RPLACD(FIRST,0);
   END;
   END OF CASE STMT;
END OF INNER BLOCK;
END OF PROCEDURE;
INTEGER QM,QN;
ARRAY QA[0:0];
PROCEDURE NAME(MFID,FID); VALUE MFID,FID; REAL MFID,FID;
    BEGIN INTEGER I; FILL NEWDISK WITH MFID,FID;
    FOR I:=0 STEP 1 UNTIL MBUFF DO
        FILL POINTERS[I] WITH MFID,FID;
    FILL ESTABLISH WITH MFID,FID;
    SETPOINTERNAMES
    END;
PROCEDURE SEQUENTIAL(UNIT); VALUE UNIT; INTEGER UNIT;
    MEMORY(11,UNIT,QA,QN,QM);
INTEGER PROCEDURE CONTENTS(UNIT,N,AR); VALUE UNIT,N;
    INTEGER UNIT,N; ARRAY AR[0];
    BEGIN
    MEMORY(4,UNIT,AR,N,QM); CONTENTS:=QM;
    END;
PROCEDURE DELETE1(UNIT,N); VALUE UNIT,N; INTEGER UNIT,N;
    MEMORY(6,UNIT,QA,N,QM);
INTEGER PROCEDURE SEARCHORD(UNIT,REC,LOC,M); VALUE UNIT,M;
    INTEGER UNIT,LOC,M; ARRAY REC[0];
    BEGIN LOC:=1;
    MEMORY(7,UNIT,REC,LOC,M);
    SEARCHORD:=M;
    END;
PROCEDURE STOREORD(UNIT,REC,N); VALUE UNIT,N; INTEGER UNIT,N;
    ARRAY REC[0];
    MEMORY(5,UNIT,REC,N,QM);
PROCEDURE STOREORDR(UNIT,REC,N); VALUE UNIT,N; INTEGER UNIT,N;
    ARRAY REC[0];
    MEMORY(2,UNIT,REC,N,QM);
BOOLEAN PROCEDURE MAINTENANCE;
    BEGIN MEMORY(9,0,QA,QN,QM); MAINTENANCE:=QN=1
    END;
PROCEDURE WRAPUP; MEMORY(10,0,QA,QN,QM);
INTEGER PROCEDURE STORESEQ(UNIT,REC,N); VALUE UNIT,N; INTEGER UNIT, N;
    ARRAY REC[0];
    BEGIN
    MEMORY(5,UNIT,REC,QN,N); STORESEQ:=QN;
    END;
PROCEDURE DELETEN(UNIT,N,M); VALUE UNIT,N,M; INTEGER UNIT,N,M;
    BEGIN M:=M-N;
    DO MEMORY(6,UNIT,QA,N,QM) UNTIL M:=M-1 LSS 0;
    END;
INTEGER PROCEDURE NEXTUNIT;
    BEGIN MEMORY(1,0,QA,QN,1); NEXTUNIT:=QN
    END;
INTEGER PROCEDURE SIZE(UNIT); VALUE UNIT; INTEGER UNIT;
    BEGIN MEMORY(3,UNIT,QA,QN,QM); SIZE:=QM
    END;
PROCEDURE ALLOCATE(J,FACTOR); VALUE J,FACTOR; INTEGER J;
    REAL FACTOR;
    BEGIN
    QN:=ENTIER( ABS( (FACTOR × 100) MOD 101));
    MEMORY(12,0,QA,QN,J)
    END;
PROCEDURE RELEASEUNIT(UNIT); VALUE UNIT; INTEGER UNIT;
    MEMORY(13,UNIT,QA,QN,QM);
DEFINE
    ALLOWQUESIZE=4#,
    ACOUNT=ACCUM[0].[1:11]#,
    DATADESC=[1:1]#,
    SCALAR=[4:1]#,
    NAMED=[3:1]#,
    CHRMODE=[5:1]#,
    CHECKT=5#,    % NUMBER OF TIMES THRU EXECUTE BEFORE CHECK
    CCIF=18:36:12#,
    CDID=1:43:5#,
    CSPF=30:30:18#,
    CRF=24:42:6#,
    CLOCF=6:30:18#,
    PF=[1:17]#,
    XEQMODE=1#,
    FUNCMODE=2#,
    CALCMODE=0#,
    INPUTMODE=3#,
    ERRORMODE=4#,
    FUNCTION=1#,
    CURRENTMODE = PSRM[0]#,
    VARIABLES = PSRM[1]#,
    VARSIZE = PSRM[2]#,
FUNCPOINTER = PSRM[3]#,
FUNCSEQ = PSRM[4]#,
CURLINE = PSRM[5]#,
STACKBASE = PSRM[6]#,
INCREMENT=STACKBASE#,   %FUNCMODE/CALCMODE
SYMBASE = PSRM[7]#,
FUNCSIZE=SYMBASE#,      %FUNCMODE/CALCMODE
USERMASK = PSRM[8]#,
SEED = PSRM[10]#,
ORIGIN = PSRM[11]#,
FUZZ = PSRM[12]#,
FSTART=9#,  %PSR[9] IS WHERE NAME OF CURRENTLY EDITED FCN GOES
PSRSIZE = 13#,
PSR = PSRM[*]#,
WF=[18:8]#,
WDSPERREC=10#,
WDSPERBLK=30#,
NAREAS=10#,
SIZEAREAS=210#,
LIBF1=[6:15]#,
LIBF2=[22:16]#,
LIBF3=[38:10]#,
LIBSPACES=1#,
IDENT=RESULT=1#,
SPECIAL=RESULT=3#,
NUMERIC=RESULT=2#,
REPLACELOC=0#,
REPLACEV=4#,
SPF=[30:18]#,
RF=[24:6]#,
DID=[1:5]#,
XRF=[12:18]#,
DDPNSW=30#, % DATA DESC PRESENT NAMED SCALAR WORD
DDNNVW=20#, %DATA DESC NON-PRES NAMED VECTOR WORD
DDNUVW=16#, %DATA DESC NONPRES..(POINTS INTO SYM TAB FOR LOCALS)
DDPUVW=24#, % DATA DESC PRESENT UNNAMED VECTOR WORD
DDNNSW=22#, % DATA DESC NON-PRES NAMED SCALAR WORD
PDC=10#, % PROG DESC CALC MODE
INTO=0#,
DDPUSW=26#,     % DATA DESC PRESENT UNNAMED SCALAR WORD (MODE)
DDPUSC=27#,     % DATA DESC PRESENT UNNAMED SCALAR CHR
DDPUVC=25#,     % DATA DESC PRESENT UNNAMED VECTOR CHR
DDPNVC=29#,  %DATA DESC PRES PERMANENT VECTOR CHAR MODE
DDPNVW=28#,  %DATA DESC PRES NAMED VEC WORD (NAMED=PERMANENT)
OUTOF=1#,
NAMEDNULLV=0&7[1:45:3]#, %KLUDGE...NAMED VERSION OF NULLV
BACKP=[6:18]#,
    SCALARDATA=0#,
    ARRAYDATA=2#,
    DATATYPE=[4:1]#,
    ARRAYTYPE=[5:1]#,
    CHARARRAY=1#,
   -NUMERICARRAY=0#,
     BLOCKSIZE=30#, %#WORDS OF CONTIGUOUS DATA IN SEQUENTIAL STORE
    VARTYPE=[42:6]#,
    WS=WORKSPACE#,
DIMPTR=SPF#,
INPTR=BACKP#,
QUADIN=[18:3]#,
QUADINV=18:45:3#,
STATEVECTORSIZE=16#,
SUSPENDED=[5:1]#,
SUSPENDVAR=[2:1]#,
CTYPEF=3:45:3#,
CSUSVAR=2:47:1#,
CNAMED=3:47:1#,
MAXWORDSTORE=3960#, %APL PREVENTS CREATION OF ARRAYS BIGGER THAN
                    %3960 ELEMENTS.  THIS NUMBER IS THE PRODUCT OF
                    %4,(NUMBER OF POINTERS TO SEQUENTIAL STORE
                    %BLOCKS THAT ARE STORED IN ONE WORD)
                    %30, (BLOCKSIZE),
                    %AND 33, (SIZE OF ARRAY USED TO STORE THESE
                    %POINTERS IN GETARRAY, MOVEARRAY, AND
                    %RELEASEARRAY).  SUBSCRIPTS ALLOWS 8x3960
                    %ELEMENTS IF THEY ARE CHARACTERS,
                    %HOWEVER, SP WILL GET FULL BEFORE THAT SINCE
                    %BIGGEST SP SIZE IS CURRENTLY 3584
MAXBUFFSIZE=30#,
MAXHEADERARGS=30#,
BUFFERSIZE=BUFFSIZE#,
LINEBUFFER=LINEBUFF#,
LINEBUFF = OUTBUFF[*]#,
APPENDTOBUFFER=APPENDTOBUFF#,
FOUND=TARRAY[0]#,
EOB=TARRAY[1]#,
MANT=TARRAY[2]#,
MANTLEN=TARRAY[3]#,
FRAC=TARRAY[4]#,
FRACLEN=TARRAY[5]#,
POWER=TARRAY[6]#,
POWERLEN=TARRAY[7]#,
MANTSIGN=TARRAY[8]#,
TABSIZE = 43#,
LOGINCODES=1#,
LOGINPHRASE=2#,
LIBRARY=1#,
WORKSPACEUNIT=2#,
RTPAREN=9#,
MASTERMODE=USERMASK.[1:1]#,
EDITOG=USERMASK.[2:1]#,
POLBUG=USERMASK.[3:1]#,
FPTF=9#,       % FUNCTION POINTER FIELD (STARTS AT CHR POS 9)
FSQF=11#,      % FUNCTION SEQNTL FIELD
FFL=2#,        % FUNCTION FIELD LENGTH (2 CHR POSITIONS)
CRETURN=3:47:1#,
RETURNVALUE=[3:1]#,
CNUMBERARGS=4:46:2#,
NUMBERARGS=[4:2]#,
RETURNVAL=1#,
NOSYNTAX=USERMASK.[4:1]#,
LINESIZE=USERMASK.[41:7]#,
DIGITS=USERMASK.[37:4]#,
SUSPENSION=USERMASK.SUSPENDED#,
SAVEDWS=USERMASK.[7:1]#,
DELTOG=USERMASK.[6:1]#,
DELCHR="$"#, %USED IN DELPRESENT (IN FUNCTIONHANDLER)
MAXMESS=27#,
USERTOP=21#,
MARGINSIZE=6#,
LFTBRACKET=SPECIAL AND ACCUM[0]=11#,
QUADV=SPECIAL AND ACCUM[0]=10#,
QUOTEV=ACCUM[0]=20#,
EXPANDV=38#,
SLASHV=6#,
GOTOV=5#,
DOTV=17#,
ROTV=37#,
RGTBRACKET=SPECIAL AND ACCUM[0]=12#,
DELV=SPECIAL AND ACCUM[0]=13#,
PLUS         = SPECIAL AND ACCUM[0] = 48#,
MINUS        = SPECIAL AND ACCUM[0] = 49#,
NEGATIVE      = SPECIAL AND ACCUM[0] = 51#,
TIMES        = SPECIAL AND ACCUM[0] = 50#,
LOGS         = SPECIAL AND ACCUM[0] = 54#,
SORTUP       = SPECIAL AND ACCUM[0] = 55#,
SORTDN       = SPECIAL AND ACCUM[0] = 56#,
NAND         = SPECIAL AND ACCUM[0] = 58#,
NOR          = SPECIAL AND ACCUM[0] = 59#,
TAKE         = SPECIAL AND ACCUM[0] = 60#,
DROPIT       = SPECIAL AND ACCUM[0] = 61#,
LFTARROW     = SPECIAL AND ACCUM[0] = 04#,
TRANS        = SPECIAL AND ACCUM[0] = 05#,
SLASH        = SPECIAL AND ACCUM[0] = 06#,
INTDIVIDE    = SPECIAL AND ACCUM[0] = 07#,
LFTPAREN     = SPECIAL AND ACCUM[0] = 08#,
RGTPAREN     = SPECIAL AND ACCUM[0] = 09#,
QUOTEQUAD    = SPECIAL AND ACCUM[0] = 14#,
SEMICOLON    = SPECIAL AND ACCUM[0] = 15#,
COMMA        = SPECIAL AND ACCUM[0] = 16#,
DOT          = SPECIAL AND ACCUM[0] = 17#,
STAR         = SPECIAL AND ACCUM[0] = 18#,
AT           = SPECIAL AND ACCUM[0] = 19#,
QUOTE        = SPECIAL AND ACCUM[0] = 20#,
BOOLAND      = SPECIAL AND ACCUM[0] = 21#,
BOOLOR       = SPECIAL AND ACCUM[0] = 22#,
BOOLNOT      = SPECIAL AND ACCUM[0] = 23#,
LESSTHAN     = SPECIAL AND ACCUM[0] = 24#,
LESSEQ       = SPECIAL AND ACCUM[0] = 25#,
EQUAL        = SPECIAL AND ACCUM[0] = 26#,
GRTEQ        = SPECIAL AND ACCUM[0] = 27#,
GREATER      = SPECIAL AND ACCUM[0] = 28#,
NOTEQ        = SPECIAL AND ACCUM[0] = 29#,
CEILING      = SPECIAL AND ACCUM[0] = 30#,
FLOOR        = SPECIAL AND ACCUM[0] = 31#,
STICK        = SPECIAL AND ACCUM[0] = 32#,
EPSILON      = SPECIAL AND ACCUM[0] = 33#,
RHO          = SPECIAL AND ACCUM[0] = 34#,
IOTA         = SPECIAL AND ACCUM[0] = 35#,
TRACE        = SPECIAL AND ACCUM[0] = 36#,
PHI          = SPECIAL AND ACCUM[0] = 37#,
EXPAND       = SPECIAL AND ACCUM[0] = 38#,
BASVAL       = SPECIAL AND ACCUM[0] = 39#,
EXCLAMATION  = SPECIAL AND ACCUM[0] = 40#,
MINUSLASH    = SPECIAL AND ACCUM[0] = 41#,
QUESTION     = SPECIAL AND ACCUM[0] = 42#,
OSLASH       = SPECIAL AND ACCUM[0] = 43#,
TAU          = SPECIAL AND ACCUM[0] = 44#,
CIRCLE       = SPECIAL AND ACCUM[0] = 45#,
LOCKIT        =IDENT AND ACCUM[0]="4LOCK  "#,
COLON        = SPECIAL AND ACCUM[0] = 47#,
QUADLFTARROW=51#,
REDUCT=52#,
ROTATE=53#,
SCANV=57#,
LINEBUFFSIZE=17#,
MAXPOLISH=100#, MESSIZE=10#,
MAXCONSTANT=30#,
MAXMEMACCESSES=3584#,    %MAXSPROWS x SPRSIZE
MAXSYMBOL=30#,
MAXSPROWS=28#,
TYPEFIELD=[3:3]#,
OPTYPE=[1:2]#,
LOCFIELD=BACKP#,
ADDRFIELD=SPF#,
SYMTYPE=[3:3]#,
OPERAND=5#,
CONSTANT=2#,
OPERATOR=3#,
LOCALVAR=4#,
SYMTABSIZE=1#,
LFTPARENV=8#,
RGTPARENV=9#,
LFTBRACKETV=11#,
RGTBRACKETV=12#,
SEMICOLONV=15#,
QUAD=10#,
QQUAD=14#,
LFTARROWV=4#,
SORTUPV=55#,
SORTDNV=56#,
ALPHALABEL=1#,
NUMERICLABEL=2#,
NEXTLINE=0#,
ERRORCOND=3#,
PRESENCE=[2:1]#,
CHANGE=[1:1]#,
XEQ=1#,
CLEARCORE=2#,
WRITECORE=3#,
%%%
%%%
XEQUTE=1#,
SLICE=120#,     %TIME SLICE IN 60THS OF A SECOND
ALLOC=2#,
WRITEBACK=3#,
LOOKATSTACK=5#,

LEN=[1:23]#,
NEXT=[24:24]#,
LOC=L.[30:11],L.[41:7]#,
NOC=N.[30:11],N.[41:7]#,
MOC=M.[30:11],M.[41:7]#,
SPRSIZE=128#, % SP ROW SIZE
NILADIC=0#,
MONADIC=1#,
DYADIC=2#,
TRIADIC=3#,
    DEPTHERROR=1#,
    DOMAINERROR=2#,
    INDEXERROR=4#,
    LABELERROR=5#,
    LENGTHERROR=6#,
    NONCEERROR=7#,
    RANKERROR=8#,
    SYNTAXERROR=9#,
    SYSTEMERROR=10#,
    VALUEERROR=11#,
    SPERROR=12#,
    KITEERROR=13#,
    STREAMBASE=59823125#,
    APLOGGED=[10:1]#,
    APLHEADING=[11:1]#,
    CSTATION = STATION#,
    CAPLOGGED=10:47:1#,
    CAPLHEADING=11:47:1#,
    APLCODE = STATIONPARAMS#,


    SPECMODE = BOUNDARY.[1:3]#,
    DISPLAYING=1#,
    EDITING=2#,
    DELETING=3#,
    RESEQUENCING=4#,
    LOWER = BOUNDARY.[4:22]#,
    UPPER = BOUNDARY.[26:22]#,
    OLDBUFFER = OLDINPBUFFER[*]#,

    ENDEFINES=#;
    REAL ADDRESS, ABSOLUTEADDRESS,
    LADDRESS;
    BOOLEAN LINETOG; %GO TO NEXT LINE IF TRUE WHEN WRITING OUT
    INTEGER BUFFSIZE,ITEMCOUNT,RESULT,
    LOGINSIZE,
%%%
    ERR,
    NROWS,
%%%
    CUSER;
LABEL ENDOFJOB,TRYAGAIN;
REAL GT1,GT2,GT3;
DEFINE LINE=PRINT#;
SAVE ARRAY BUFFER[0:MAXBUFFSIZE];
ARRAY TARRAY[0:8],
    COMMENT PROGRAM STATE REGISTER;
    PSRM[0:PSRSIZE],
    OLDINPBUFFER[0:MAXBUFFSIZE],
    SP[0:27, 0:SPRSIZE-1],
    IDTABLE[0:TABSIZE],
    MESSTAB[0:MAXMESS],
    JIGGLE[0:0],
    SCR[0:2],
    CORRESPONDENCE[0:7],
    ACCUM[0:MAXBUFFSIZE];
    DEFINE OUTBUFFSIZE=29#,CLOGGED=7:47:1#,STU=15:9:9#;
    ARRAY OUTBUFF[0:OUTBUFFSIZE];
    ALPHA STATION, JOBNUM,  STATIONPARAMS, BOUNDARY;
    INTEGER CHRCOUNT, WORKSPACE;

STREAM PROCEDURE INITBUFF(B,BUFFSIZE); VALUE BUFFSIZE;
    BEGIN
    DI←B; BUFFSIZE(DS←8LIT" "); DS←LIT"←";
    END;
STREAM PROCEDURE TRANSFER(A,AS,B,BS,L); VALUE AS,BS,L;
    BEGIN LOCAL T,U,V;
    SI:=LOC AS; DI:=LOC T; DI:=DI+1; DS:=7CHR;
    SI:=LOC BS; DI:=LOC U; DI:=DI+1; DS:=7CHR;
    SI:=LOC L; DI:=LOC V; DI:=DI+1; DS:=7CHR;
    SI:=A; T(2(SI:=SI+32)); SI:=SI+AS;
    DI:=B; U(2(DI:=DI+32)); DI:=DI+BS;
    V(2(DS:=32CHR)); DS:=L CHR;
    END;
REAL PROCEDURE NUMBER; FORWARD; %LINE 111500
BOOLEAN PROCEDURE SCAN;
 BEGIN
REAL STREAM PROCEDURE GNC(ADDR,ACC); VALUE ADDR;
 BEGIN SI:=ADDR; DI:=ACC; DI:=DI+7; DS:=CHR; GNC:=SI;
 DI:=ACC; SKIP DB; DS:=SET; END OF GNC;
REAL STREAM PROCEDURE RESWD(TAB,BUF,ADDR,EOB,FOUND,K);
    VALUE ADDR,K;
    BEGIN
    LOCAL T,TSI,TDI;
    LABEL TRY,L,KEEPGOING,FINIS,RESTORE;
    LABEL NUMBERFOUND;
    DI:=EOB; DS:=8LIT"0"; DI:=FOUND; DS:=8LIT"0";
    SI:=ADDR;
    L: IF SC NEQ " " THEN GO TO KEEPGOING;
        SI:=SI+1;
        GO TO L;
    KEEPGOING:
    RESWD:=SI;
    ADDR:=SI;
    IF SC GEQ "0" THEN IF SC LEQ "9" THEN GO TO NUMBERFOUND;
```

```
        IF SC="#" THEN GO TO NUMBERFOUND;
        IF SC="@" THEN GO TO NUMBERFOUND;
        IF SC="." THEN
               BEGIN SI:=SI+1;
               IF SC GEQ "0" THEN IF SC LEQ "9" THEN
               GO TO NUMBERFOUND; SI:=SI-1;
               END;
    DI:=LOC T; DS:=2RESET; DS:=2SET; DS:=2RESET;
    DI:=LOC T;
    IF SC=DC THEN
        BEGIN DI:=EOB; DI:=DI+7; DS:=LIT"1";
        GO TO FINIS
        END;
    SI:=TAB; TSI:=SI;
    TRY:
    IF SC="O" THEN
        BEGIN SI:=ADDR;
        IF SC=ALPHA THEN
        IF SC GEQ"0" THEN
            IF SC LEQ "9" THEN
NUMBERFOUND:
            TALLY:=2 ELSE TALLY := 0
        ELSE TALLY:=1
        ELSE TALLY:=3;
        T:=TALLY; SI:=LOC T; SI:=SI+7; DI:=FOUND; DI:=DI+7;
        DS:=CHR; GO FINIS;
        END;
    DI:=LOC T; DI:=DI+7; DS:=CHR;
    DI:=ADDR;
    IF T SC=DC THEN
        BEGIN
        TSI:=SI; TDI:=DI; SI:=SI-1;
        IF SC=ALPHA THEN
            BEGIN DI:=DI+16; SI:=TDI;
            IF SC NEQ " " THEN IF SC =ALPHA THEN ;
            END;
        SI:=TSI;
        END ELSE GO TO RESTORE;
    IF TOGGLE THEN
    RESTORE:
        BEGIN SI:=SI+K; DI:=ADDR; GO TO TRY
        END;
    DI:=FOUND; DS:=K OCT;
    DI:=TDI; RESWD:=DI;
    FINIS:
    END;
REAL STREAM PROCEDURE ACCUMULATE(ACC,EOB,ADDR); VALUE ADDR;
   BEGIN LOCAL T; LABEL EOBL,E,ON,L;
   DI:=ACC; 9(DS:=8LIT" ");
   DI:=EOB; DS:=8LIT"0"; SI:=ADDR; DI:=LOC T; SKIP 2 DB;
   DS:=2SET; DI:=LOC T;
   63(IF SC=ALPHA THEN TALLY:=TALLY+1 ELSE JUMP OUT TO E;
   SI:=SI+1);
   L:  IF SC=ALPHA THEN BEGIN SI:=SI+1; GO L END ELSE GO ON;
   IF SC=" " THEN GO ON;
   E:  IF SC = DC THEN ;
      SI:=SI-1; IF TOGGLE THEN GO TO EOBL ELSE GO ON;
   EOBL:  DI:=EOB; DI:=DI+7; DS:=LIT"1";
   ON:  ACCUMULATE:=SI; DI:=ACC; T:=TALLY; SI:=LOC T; SI:=SI+6;
   DS:=2CHR; SI:=ADDR; DS:=T CHR;
   END OF ACCUMULATE;
BOOLEAN STREAM PROCEDURE ARROW(ADDR,I); VALUE ADDR,I;
   BEGIN SI:=ADDR; SI:=SI-1; DI:=LOC I; DI:=DI+7;
   IF SC=DC THEN TALLY:=1; ARROW :=TALLY
   END OF ARROW;
  IF NOT BOOLEAN(EOB) THEN BEGIN
     LADDRESS:=ADDRESS;
     ADDRESS:=RESWD(IDTABLE,BUFFER,ADDRESS,EOB,FOUND,2);
     IF RESULT:=FOUND NEQ 0 THEN BEGIN
        IF RESULT=1 THEN ADDRESS:=ACCUMULATE(ACCUM,EOB,ADDRESS)
        ELSE IF RESULT=2 THEN ACCUM[0]:=NUMBER
        ELSE IF RESULT=3 THEN ADDRESS:=GNC(ADDRESS,ACCUM)
        ELSE BEGIN ACCUM[0]:=RESULT; RESULT:=3 END;
        ITEMCOUNT:=ITEMCOUNT+1;
        SCAN:=TRUE;
        IF ARROW(ADDRESS,31) THEN
           BEGIN EOB:=1; SCAN:=FALSE END;
        END ELSE EOB:=1;
     END;
   END OF THE SCAN PROCEDURE;
PROCEDURE FORMROW(CC,BL,A,S,N); VALUE CC,BL,S,N;
    INTEGER CC,BL,S,N; ARRAY A[0]; FORWARD
    ;
```

```
PROCEDURE INDENT(R); VALUE R; REAL R; FORWARD;
PROCEDURE TERPRINT; FORWARD;
PROCEDURE PROCESS(MODE);VALUE MODE;INTEGER MODE; FORWARD;
REAL STREAM PROCEDURE ABSADDR(A);
    BEGIN SI:=A; ABSADDR:=SI
    END;
BOOLEAN PROCEDURE LIBRARIAN(MFID,FID); VALUE MFID,FID;
    REAL MFID,FID;
    BEGIN
    REAL ARRAY A[0:6]; FILE DF DISK(1,1);
    REAL T;
    COMMENT LIBRARIAN IS TRUE IF MFID/FID IS PRESENT ON DISK;
     FILL DF WITH MFID,FID;
    SEARCH(DF,A[*]);
    LIBRARIAN:=
        A[0]≠-1;
    END;
FILE SPO 11(1,3);
PROCEDURE SPOUT(K); VALUE K; INTEGER K;
    BEGIN FORMAT ERRF("APL ERROR:",I8,A1);
    WRITE(SPO,ERRF,K,31);
    END;
PROCEDURE INITIALIZETABLE;
    BEGIN DEFINE STARTSEGMENT= #;
    INTEGER I;
    LADDRESS:=
    ABSOLUTEADDRESS:=ABSADDR(BUFFER);
    BIGGEST := REAL(NOT FALSE) & 0[1:46:2];
    NULLV := 0 & 3[1:46:2];
    STATUSWORD←REAL(BOOLEAN(STATUSWORD) OR BOOLEAN(1));
    JOBNUM←TIME(-1);
    STATION←0&1[CLOGGED]&STATUSWORD[STU];
    FILL JIGGLE[*] WITH OCT5757575757575737;%CARRIAGE RETURNS LEFT ARROW
    FILL IDTABLE[*] WITH
    "1+481-49", "1&501%07", "1.171@19", "1#411(08",
    "1)091/06", "3XEQ623L", "OG541;15", OCT0333777601040177,
            %LAST IN ABOVE LINE IS REALLY 3["]141"
    "202:=042", "[]101[11", "1]123AND", "212OR223",
    "NOT233LS", "S243LEQ2", "53GEQ273", "GTR283NE", "Q292=:05",
    "2GO051=2", "63MAX304", "CEIL303F", "LR313MIN",
    "314RESD3","23ABS323","RHO341*1","84IOTA35",
    "1×384RND", "M425TRAN", "S431$133", "PHI374FA",
    "CT404COM", "B406CIRC", "LE456SOR", "TUP556SO",
    "RTDN561:", "474NAND5", "83NOR594", "TAKE604D",
    "ROP613RE", "P446BASV", "AL393EPS", "331,1600";
  COMMENT IDTABLE IS TABLE OF RESERVED WORDS AND SPECIAL SYMBOLS.
    FORMAT IS NUMBER OF CHARACTERS IN SYMBOL, FOLLOWED BY SYMBOL
    ITSELF, FOLLOWED BY A TWO-DIGIT DECIMAL CODE WHICH APL USES
    FOR THE RESERVED WORD--LIKE IN THE EXECUTION CASE STATEMENT AND
    IN SYNTAX CHECKING.  FOR SCAN TO WORK, THE TWO-DIGIT CODE MUST
    BE GREATER THAN 3 AND IDTABLE MUST HAVE AT LEAST ONE "0" AT THE
    END TO MARK THE END.  TABSIZE IS THE DEFINE (LINE 30000) GIVING
    THE SIZE OF IDTABLE;
    IF STACKSIZE=0 THEN STACKSIZE:=100 ELSE
        IF STACKSIZE GTR 1022 THEN STACKSIZE:=1022;
    BUFFSIZE:=MAXBUFFSIZE;
    LINETOG := TRUE; %USUALLY GO TO NEXT LINE WHEN WRITING OUT

    INITBUFF(OUTBUFF, 10);
    INITBUFF(BUFFER,BUFFSIZE);
    NROWS:=-1;
    NAME(LIBJOB,TIME(-1));
    FILL MESSTAB[*] WITH
    "4SAVE  ",
    "4LOAD  ",
    "5CLEAR ",
    "4COPY  ",
    "4VARS  ",
    "3FNS   ",
    "6LOGGED",
    "3MSG   ",
    "5WIDTH ",
    "3OPR   ",
    "6DIGITS",
    "3OFF   ",
    "6ORIGIN",
    "4SEED  ",
    "4FUZZ  ",
    "3SYN   ",
    "5NOSYN ",
    "5STORE ",
    "5ABORT ",
    "2SI    ",
    "3SIV   ",
    "5ERASE ",
    %--------------MASTERMODE BELOW HERE...(SEE USERTOP)--------
    "6ASSIGN",
    "6DELETE",
    "4LIST  ",
    "5DEBUG ",
    "5FILES ";

    IF LIBSIZE=-1 THEN
    BEGIN LIBSIZE←1;GTA[0]←"        ";STOREORD(LIBRARY,GTA,0);WRAPUP;
    END ELSE BEGIN LIBSIZE←SIZE(LIBRARY);
              FOR I←1 STEP 1 UNTIL LIBSIZE-1 DO
                BEGIN GT1←CONTENTS(LIBRARY,I,ACCUM);
                IF NOT LIBRARIAN(ACCUM[0],TIME(-1)) THEN
                BEGIN DELETE1(LIBRARY,I);LIBSIZE←LIBSIZE-1;END;
    IF (LOGINSIZE:=SIZE(LOGINCODES)=0) THEN
                END;
              END;
    FILL CORRESPONDENCE[*] WITH
      OCT1111111111110311,
      OCT1111111111111111,
      OCT1104111121221113,
      OCT2014151617100706,
      OCT1111111111111112,
      OCT1111111111111100,
      OCT0201111111251111,
      OCT2324111111111111;
    COMMENT  CORRESPONDENCE GIVES THE CORRESPONDENCE BETWEEN THE
      APL CODES FOR DYADIC SCALAR OPERATORS (EXCEPT CIRCLE) AND
      THEIR POSITIONS IN THE "CASE STATEMENT" IN "OPERATION".
      E.G. APL CODE 7 IS "OPERATION" CODE 3 IN OCTAL (FOR DIVIDE).
      IF N-TH CHARACTER IN CORRESPONDENCE IS OCTAL 11, THEN N
      IS NOT AN APL CODE FOR A DYADIC SCALAR OPERATOR.  CHARACTER
      COUNT STARTS AT 1 FOR FIRST CHARACTER. TO MAKE IT COME OUT
      RIGHT, STREAM PROCEDURE GETOP IS ACTUALLY CALLED WITH APL
      OPERATION CODE MINUS 1;
    END;
REAL STREAM PROCEDURE CONV(ADDR,N);
  VALUE N,ADDR;
  BEGIN SI:=ADDR;
  DI:=LOC CONV;
  DS:=N OCT; END;
REAL STREAM PROCEDURE BUMP(ADDR,N); VALUE ADDR,N;
  BEGIN SI:=ADDR; SI:=SI+N; BUMP:=SI; END;
REAL PROCEDURE NUMBER;
  BEGIN REAL NCHR;
  LABEL GETFRAC,GETPOWER,QUIT,KITE;
  MONITOR EXPOVR;
  REAL PROCEDURE INTCON(COUNT); VALUE COUNT;
       REAL COUNT;
    BEGIN REAL TLO,THI,T; INTEGER N;
    BOOLEAN DPTOG; DEFINE ADDR=ADDRESS#;
    COMMENT:   VALUE OF INTCON IS THE CONVERSION OF AN INTEGER
               CONSISTING OF COUNT NUMERICAL CHARACTERS STARTING
               AT THE CHARACTER ADDRESS.  ADDRESS IS SET TO POINT
               TO THE NEXT CHARACTER DURING INTCON;
    DPTOG:=COUNT GTR 8;
    THI:=T:=CONV(ADDR,N:=COUNT MOD 8);
    ADDR:=BUMP(ADDR,N);
    COUNT:=COUNT DIV 8;
    FOR N:=1 STEP 1 UNTIL COUNT DO BEGIN
         IF DPTOG THEN BEGIN
         DOUBLE(THI,TLO,100000000,0,0,×,CONV(ADDR,8),
                0,+,:=,THI,TLO);
           T:=THI
       END ELSE T:=T×100000000 + CONV(ADDR,8);
         ADDR:=BUMP(ADDR,8); END;
    INTCON:=T;
    END OF INTCON;
  INTEGER STREAM PROCEDURE SUBSCAN(ADDR,NEXT); VALUE ADDR;
    BEGIN SI:=ADDR;
    63(IF SC GEQ "0" THEN
      IF SC LEQ "9" THEN BEGIN SI:=SI+1; TALLY:=TALLY+1;
              END ELSE JUMP OUT);
    DI:=NEXT; DI:=DI+7; DS:=1 CHR; SUBSCAN:=TALLY;
    END;
  COMMENT--VALUE OF SUBSCAN IS NUMBER OF NUMERIC CHARACTERS
   FOUND. NEXT CONTAINS THE FIRST NON-NUMERIC CHARACTER;
  EXPOVR:=KITE;
  MANTSIGN:=1;
  MANT:=MANTLEN:=POWER:=POWERLEN:=FRAC:=FRACLEN:=0;
  MANTLEN:=SUBSCAN(ADDRESS,NCHR);
  IF MANTLEN=0 AND NCHR="#" THEN BEGIN
     MANTSIGN:=-1;
     ADDRESS:=BUMP(ADDRESS,1);
     MANTLEN:=SUBSCAN(ADDRESS,NCHR); END;
  IF MANTLEN=0 THEN BEGIN ADDRESS:=BUMP(ADDRESS,1);
     IF NCHR="." THEN GO TO GETFRAC
        ELSE IF NCHR="@" OR NCHR="E" THEN GO TO GETPOWER
            ELSE BEGIN ERR:=SYNTAXERROR;
                       GO TO QUIT; END; END;
  MANT:=INTCON(MANTLEN);
  IF NCHR="." THEN BEGIN ADDRESS:=BUMP(ADDRESS,1); GO GETFRAC END;
  IF NCHR="@" OR NCHR="E" THEN BEGIN
     ADDRESS:=BUMP(ADDRESS,1); GO TO GETPOWER END;
  IF NCHR=12 THEN EOB:=1;
  GO TO QUIT;
  GETFRAC:  FRACLEN:=SUBSCAN(ADDRESS,NCHR);
    IF FRACLEN=0 THEN BEGIN ERR:=SYNTAXERROR; GO TO QUIT; END;
    FRAC:=INTCON(FRACLEN);
    IF NCHR="@" OR NCHR="E" THEN BEGIN
       ADDRESS:=BUMP(ADDRESS,1); GO TO GETPOWER; END;
    IF NCHR=12 THEN EOB:=1 ELSE
       IF NCHR="." OR NCHR="#" THEN ERR:=SYNTAXERROR;
    GO TO QUIT;
  GETPOWER:
    POWERLEN:=SUBSCAN(ADDRESS,NCHR);
    IF POWERLEN=0 THEN BEGIN
       IF NCHR="-" OR NCHR="#" THEN POWER:=-1
          ELSE IF NCHR="+" THEN POWER:=1
            ELSE BEGIN ERR:=SYNTAXERROR; GO TO QUIT; END;
       POWERLEN:=SUBSCAN(ADDRESS:=BUMP(ADDRESS,1), NCHR);
       END ELSE POWER:=1;
    IF POWERLEN=0 THEN ERR:=SYNTAXERROR
          ELSE BEGIN
            POWER:=INTCON(POWERLEN)×POWER;
            IF NCHR="#" OR NCHR="@" OR NCHR="."
                THEN ERR:=SYNTAXERROR; END;
    GO TO QUIT;
  KITE:  ERR:=KITEERROR;
  QUIT:  IF ERR=0 THEN
    NUMBER:=IF MANTLEN+FRACLEN=0 THEN
       IF POWERLEN=0 THEN 0
       ELSE MANTSIGN×10*ENTIER(POWER)
       ELSE MANTSIGN×(MANT×10*ENTIER(POWER)
             + FRAC×10*ENTIER(POWER-FRACLEN)) ELSE EOB:=1;
  END OF NUMBER;
STREAM PROCEDURE APPENDTOBUFF(BUF,NBUF,NBLANK,A,SA,NA);
    VALUE NBUF,NBLANK,SA,NA;
    BEGIN LOCAL T;
    LOCAL TSI,TDI;
    SI:=LOC NBUF; DI:=LOC T; DI:=DI+1; DS:=7CHR;
    DI:=BUF; T(2(DI:=DI+32)); DI:=DI+NBUF;
    NBLANK(DS:=LIT" "); TDI:=DI;
    SI:=LOC SA; DI:=LOC T; DI:=DI+1; DS:=7CHR;
    SI:=A; T(2(SI:=SI+32)); SI:=SI+SA;
    TSI:=SI; SI:=LOC NA; DI:=LOC T; DI:=DI+1; DS:=7CHR;
    SI:=TSI; DI:=TDI; T(2(DS:=32CHR)); DS:=NA CHR
    END;
PROCEDURE TERPRINT;
    BEGIN LABEL BK;
STREAM PROCEDURE FINISHBUFF(BUF,N,TER);VALUE N,TER;
    BEGIN LOCAL T;
    SI:=LOC TER;SI:=SI+7;IF SC="1" THEN;
    SI:=LOC N; DI:=LOC T; DI:=DI+1; DS:=7 CHR;
    DI:=BUF; T(2(DI:=DI+32));DI:=DI+N;
    IF TOGGLE THEN DS:=2 LIT"≤≠"; %CARRIAGE RETURN/LINE FEED
    DS:=RESET;DS:=5 SET; %END OF MESSAGE LEFT ARROW
    END OF FINISHBUFF;
 IF CHRCOUNT NEQ 0 THEN BEGIN
    FINISHBUFF(OUTBUFF,CHRCOUNT,LINETOG);
    CHRCOUNT:=0;
    IF LINETOG THEN
    WRITE(TWXOUT,9,OUTBUFF[*])[BK:BK] ELSE
    WRITE(TWXOUT[STOP],9,OUTBUFF[*])[BK:BK];
    INITBUFF(OUTBUFF, 10);
    END;
    IF FALSE THEN
BK: IF CURRENTMODE=XEQMODE THEN BREAKFLAG:=TRUE;
 END OF TERPRINT;
PROCEDURE FORMWD(CC,WD); VALUE CC,WD; REAL WD; INTEGER CC;
    BEGIN
    INTEGER I,K,L;
    COMMENT CC=-1 STAY ON LINE, OUTPUT, DON"T GO TO NEXT LINE
    COMMENT CC=0 STAY ON THIS LINE, MORE TO COME.
```

```
             CC=1 STAY ON THIS LINE BUT TERMINATE PRINT.                00257000
             CC=2 SKIP TO NEXT LINE - MORE TO COME.                     00258000
             CC=3 SKIP TO NEXT LINE - TERMINATE PRINT.;                 00259000
     REAL STREAM PROCEDURE OCTAL(I); VALUE I;                           00260000
         BEGIN SI:=LOC I; DI:=LOC OCTAL; DS:=8OCT                       00261000
         END;                                                           00262000
     IF L:=LINESIZE LEQ 9 OR L GTR 72 THEN L:=72; K:=2;                 00263000
     IF CC GTR 1 AND CHRCOUNT GTR 0THEN TERPRINT;                       00264000
     IF CHRCOUNT+(I:=OCTAL(WD.[1:11])) GTR L THEN                       00265000
                                                                        00266000
         BEGIN APPENDTOBUFF(LINEBUFFER,CHRCOUNT,                        00267000
         0,WD,2,K:=L-CHRCOUNT);                                         00268000
     CHRCOUNT:=L; TERPRINT;                                             00269000
                                                                        00270000
         I:=I-K;                                                        00271000
                                                                        00272000
         END;                                                           00273000
     APPENDTOBUFF(LINEBUFFER,CHRCOUNT,0,WD,K,I);                        00274000
                                                                        00274900
     CHRCOUNT:=CHRCOUNT+I;                                              00275000
     IF BOOLEAN(CC) THEN                                                00276000
        IF CC=-1 THEN BEGIN LINETOG:=FALSE;                             00276010
                          TERPRINT; LINETOG:=TRUE                       00276020
                  END ELSE TERPRINT;                                    00276030
     END;                                                               00277000
 BOOLEAN PROCEDURE FUNCTIONHEADER(SPECS,HADDR);                         00277500
     ARRAY SPECS[0]; REAL HADDR; FORWARD;                               00277600
                                                                        00278000
                                                                        00279000
                                                                        00280000
 REAL PROCEDURE LINENUMBER(R); VALUE R; REAL R;                         00280100
     COMMENT STARTS ON 8030000;                                         00280110
     FORWARD;                                                           00280120
                                                                        00280130
 PROCEDURE INDENT(R); VALUE R; REAL R;                                  00281000
     BEGIN                                                              00281100
     INTEGER STREAM PROCEDURE FORM(A,I,K);VALUE K,I;                    00281200
         BEGIN                                                          00281300
         LOCAL T1,T2;                                                   00281400
         LABEL SHORT,L,M,FINIS;                                         00281500
         TALLY:=K; FORM:=TALLY;                                         00281600
         SI:=LOC I; DI:=LOC T1; IF 8SC=DC THEN                          00281700
             BEGIN DI:=A; K(DS:=LIT" "); GO FINIS                       00281800
             END;                                                       00281900
         SI:=LOC I; DI:=A; TALLY:=3; DS:=LIT"[";                        00282000
         IF SC GTR "0" THEN IF SC LSS "0" THEN ;                        00282100
         3(TALLY:=TALLY+1; IF TOGGLE THEN DS:=CHR ELSE                  00282200
             IF SC NEQ "0" THEN DS:=CHR ELSE                            00282300
             BEGIN TALLY:=TALLY+63; SI:=SI+1                            00282400
             END );                                                     00282500
         DS:=CHR; T1:=TALLY; TALLY:=4; SI:=SI+3;                        00282600
         4(IF SC NEQ "0" THEN JUMP OUT TO M;                            00282700
         .   TALLY:=TALLY+63; SI:=SI-1); GO TO L;                       00282800
         M:                                                             00282900
         T2:=TALLY; SI:=LOC I; SI:=SI+4; DS:=LIT"."; DS:=T2 CHR;        00283000
         TALLY:=T1; TALLY:=TALLY+T2; TALLY:=TALLY+1; T1:=TALLY;         00283100
         L:                                                             00283200
         DS:=LIT"]"; TALLY:=K;                                          00283300
         T1(TALLY:=TALLY+63; T2:=TALLY; SI:=LOC T2; SI:=SI+7;           00283400
             IF SC="0" THEN JUMP OUT TO SHORT);                         00283500
         T2(DS:=LIT" "); GO FINIS;                                      00283600
         SHORT:                                                         00283700
         TALLY:=T1; TALLY:=TALLY+1; FORM:=TALLY; DS:=LIT" ";            00283800
         FINIS:                                                         00283900
         DS:=RESET; DS:=5SET;                                           00284000
         END;                                                           00284100
    IF R LSS 0 THEN R:=LINENUMBER(-R) ELSE R:=ABS(R); % -0              00285000
    CHRCOUNT:=FORM(LINEBUFF,R,MARGINSIZE)+1                             00286000
                                                                        00286100
    END;                                                                00287000
INTEGER PROCEDURE HEADER(ADDR1,ADDR2,BUF); VALUE ADDR1,ADDR2;           00287010
  INTEGER ADDR1, ADDR2; ARRAY BUF[0];                                   00287020
  BEGIN                                                                 00287030
 INTEGER STREAM PROCEDURE HEADRR(ADDR1,ADDR2,BUF); VALUE ADDR1,         00287100
    ADDR2;                                                              00287110
    BEGIN                                                               00287120
    LOCAL C,T,TDI;                                                      00287130
    LOCAL QM,AR;                                                        00287132
    LABEL L,ENDSCAN,M,N;                                                00287140
    DI:=LOC QM; DS:=2RESET; DS:=2SET;                                   00287142
    DI:=LOC AR; DS:=RESET; DS:=5SET;                                    00287144
    DI:=BUF;                                                            00287180
    SI:=ADDR1;                                                          00287200
    L: T:=SI; TDI:=DI;                                                    00287210
    DI:=LOC QM; IF SC=DC THEN GO TO ENDSCAN;                              00287212
    DI:=LOC AR; SI:=SI-1; IF SC=DC THEN GO TO ENDSCAN;                    00287214
    SI:=LOC T; DI:=LOC ADDR2;                                             00287220
    IF 8SC=DC THEN COMMENT END OF SCAN;                                   00287230
         GO TO ENDSCAN;                                                   00287240
    SI:=T; DI:=TDI; DS:=CHR;                                              00287250
    GO TO L;                                                              00287260
    ENDSCAN:                                                              00287300
    SI:=TDI;                                                              00287310
    M:     SI:=SI-1;                                                      00287320
    IF SC=" " THEN GO TO M;                                               00287330
    SI:=SI+1;                                                             00287332
    ADDR2:=SI;                                                            00287340
    SI:=BUF;                                                              00287350
    N: T:=SI; DI:=LOC ADDR2;                                              00287360
    SI:=LOC T;                                                            00287370
    IF 8SC NEQ DC THEN                                                    00287380
         BEGIN                                                            00287390
         TALLY:=TALLY+1; TDI:=TALLY;                                      00287400
         SI:=LOC TDI; SI:=SI+7;                                           00287410
         IF SC="0" THEN                                                   00287420
              BEGIN TALLY:=C; TALLY:=TALLY+1; C:=TALLY;                   00287430
              TALLY:=0;                                                   00287440
              END;                                                        00287450
         SI:=T; SI:=SI+1; GO TO N;                                        00287460
         END;                                                             00287470
    HEADRR:=TALLY; SI:=LOC C; DI:=LOC HEADRR; SI:=SI+1; DS:=6 CHR;        00287480
    END;                                                                  00287490
  HEADER:=HEADRR(ADDR1,ADDR2,BUF);                                        00287492
  END OF PHONY HEADER;                                                    00287494
PROCEDURE STARTSCAN;                                                      00299000
    BEGIN                                                                 00300000
                                                                          00300100
                                                                          00300600
                                                                          00300700
   LADDRESS:=                                                             00301000
    ADDRESS:=ABSOLUTEADDRESS;                                             00302000
         BEGIN TERPRINT;                                                  00304000
         END;                                                             00305000
    READ(TWXIN[STOP],29,BUFFER[*]);                                       00306000
    BUFFER[30]:=0&31[1:43:5];                                             00307000
    ITEMCOUNT:=0;                                                         00312000
    EOB:=0                                                                00313000
    END;                                                                  00314000
PROCEDURE FORMROW(CC,BL,A,S,N); VALUE CC,BL,S,N; INTEGER CC,BL,           00315000
    S,N; ARRAY A[0];                                                      00316000
    COMMENT:   CC--SAME CODE AS IN FORMWD, LINE 253000                    00316010
               BL--#BLANKS TO PUT IN FRONT OF IT                          00316020
                A--ARRAY WHERE THE STUFF TO PUT ON LINE IS STORED         00316030
                S--#CHARACTERS TO SKIP AT START OF A                      00316040
                N--#CHARACTERS TO TAKE FROM A TO PUT ON OUTPUT LINE;      00316050
    BEGIN INTEGER K;                                                      00317000
    INTEGER T;                                                            00317100
    IF CC GTR 1 AND CHRCOUNT GTR 0  THEN TERPRINT;                        00318000
    IF K:=LINESIZE LEQ 9 OR K GTR 72 THEN K:=72;                          00319000
    WHILE CHRCOUNT+N+BL GTR K DO                                          00320000
         BEGIN                                                            00321000
         APPENDTOBUFFER(LINEBUFFER,CHRCOUNT,BL,A,S,T:=K-CHRCOUNT-BL);     00322000
         CHRCOUNT:=K; TERPRINT;                                           00323000
         S:=S+T; N:=N-T;                                                  00324000
         BL:=0;                                                           00325000
         END;                                                             00326000
    APPENDTOBUFFER(LINEBUFFER,CHRCOUNT,BL,A,S,N);                         00327000
                                                                          00327900
    CHRCOUNT:=CHRCOUNT+N+BL;                                              00328000
    IF BOOLEAN(CC) THEN                                                   00329000
       IF CC=-1 THEN BEGIN LINETOG:=FALSE;                                00329010
                         TERPRINT; LINETOG:=TRUE;                         00329020
                    END ELSE TERPRINT;                                    00329030
    END;                                                                  00330000
PROCEDURE NUMBERCON(R,A); VALUE R; REAL R; ARRAY A[0];                    00331000
    BEGIN FORMAT F(F24,*), G(E24,*);                                      00332000
    REAL S; DEFINE MAXIM = 10@9#;                                         00332010
                                                                          00333000
    STREAM PROCEDURE ADJUST(A,B);                                         00334000
         BEGIN LOCAL T,FRAC,MANT,T1,TSI,TDI;                              00335000
         DI:=LOC T; DI:=DI+1; T1:=DI;                                     00336000
         SI:=B; DI:=A; DI:=DI+2;                                          00337000
         24(IF SC=" " THEN SI:=SI+1 ELSE                                  00338000
              BEGIN TSI:=SI; SI:=LOC T;                                   00339000
              IF SC="1" THEN; SI:=TSI;                                    00340000
              IF TOGGLE THEN                                              00341000
```

```
                    IF SC NEQ "0" THEN                                        00342000
                        IF SC="@" THEN BEGIN                                  00343000
                            TSI:=SI; DI:=T1; DS:=LIT"1"; JUMP OUT;            00343010
                            END ELSE FRAC:=TALLY                              00344000
                        ELSE TALLY := TALLY+0                                 00345000
                ELSE                                                          00346000
            IF SC="." THEN                                                    00347000
                BEGIN MANT:=TALLY; TDI:=DI; DI:=LOC T; DS:=                   00348000
                LIT"1"; TALLY:=0;DI:=TDI;                                     00349000
                END;                                                          00350000
            TALLY:=TALLY+1; DS:=CHR                                           00351000
            END);                                                             00352000
        SI:=LOC MANT; SI:=SI+7; IF SC="0" THEN MANT:=TALLY;                   00353000
                                                                              00354000
        TALLY:=MANT; SI:=LOC FRAC; SI:=SI+7; IF SC GTR "0"                    00355000
        THEN TALLY:=TALLY+1; TALLY:=TALLY+FRAC; MANT:=TALLY;                  00356000
        SI:=T1; IF SC="1" THEN BEGIN                                          00356010
            DI:=A; DI:=DI+MANT; DI:=DI+2;                                     00356020
            SI:=TSI;  DS:=4CHR;                                               00356030
            TALLY:=TALLY+4; MANT:=TALLY; END;                                 00356040
        SI:=LOC MANT; SI:=SI+6; DI:=A; DS:=2CHR;                              00357000
        END;                                                                  00358000
    IF S:=ABS(R) GEQ MAXIM OR S LEQ 10*(-DIGITS) AND S NEQ 0 THEN             00358010
        WRITE(SCR[*],G,DIGITS,R) ELSE                                         00358020
    WRITE(SCR[*],F,DIGITS,R);                                                 00359000
    ADJUST(A,SCR)                                                             00360000
    END;                                                                      00361000
PROCEDURE STOREPSR;                                                           00361010
    BEGIN INTEGER I;                                                          00361020
    DELETE1(WORKSPACE,0);                                                     00361030
    I:=STORESEQ(WORKSPACE,PSR,PSRSIZE×8);                                     00361040
        COMMENT USED TO CALL WRAPUP;                                          00361050
    END;                                                                      00361060
PROCEDURE RESCANLINE;                                                         00361070
    BEGIN ADDRESS:=ABSOLUTEADDRESS; EOB:=0; END;                              00361072
PROCEDURE PROCESS(MODE);VALUE MODE;  INTEGER MODE; FORWARD;                   00361100
PROCEDURE MESSAGEHANDLER; FORWARD;                                            00362000
PROCEDURE FUNCTIONHANDLER; FORWARD;                                           00362100
PROCEDURE ERRORMESS(N,ADDR,R); VALUE N,ADDR,R; REAL R;                        00362105
    INTEGER N;REAL ADDR;FORWARD; COMMENT LINE 5000000;                         00362107
STREAM PROCEDURE SETFIELD(A,S,L,R); VALUE S,L,R;                              00362110
    BEGIN DI:=A; DI:=DI+S; SI:=LOC R; SI:=SI+8; L(SI:=SI-1);                  00362120
    DS:=L CHR;                                                                00362130
    END;                                                                      00362140
COMMENT:  VALUE OF GETFIELD IS L CHARACTERS, STARTING AT J-TH                 00362145
          CHARACTER OF A, RIGHT-ADJUSTED.  L MUST BE LEQ 8 AND                00362146
          J MUST BE LESS THAT 64;                                             00362147
REAL STREAM PROCEDURE GETFIELD(A,S,L); VALUE S,L;                             00362150
    BEGIN SI:=A; SI:=SI+S; DI:=LOC GETFIELD; DI:=DI+8; L(DI:=DI-1);           00362160
    DS:=L CHR;                                                                00362170
    END;                                                                      00362180
REAL PROCEDURE TOPLINE(ORD); VALUE ORD; INTEGER ORD;                          00362200
    BEGIN                                                                     00362210
    INTEGER STREAM PROCEDURE CON(A); VALUE A;                                 00362220
        BEGIN SI:=LOC A; DI:=LOC CON; DS:=8OCT END;                           00362230
        ARRAY A[0:1]; INTEGER I;                                              00362240
        I:=CONTENTS(ORD,SIZE(ORD)-1,A);                                       00362250
        TOPLINE:=CON(A[0])/10000                                              00362260
    END;                                                                      00362270
BOOLEAN PROCEDURE FUNCTIONHEADER(SPECS,HADDR);                                00500000
ARRAY SPECS[0]; REAL HADDR;                                                   00500100
BEGIN                                                                         00500150
LABEL A,B,C;                                                                  00500200
INTEGER P;                                                                    00500300
DEFINE NOTE=HADDR.[24:24]:=ADDRESS#,P8=8×P+1#;                                00500325
ERR:=0;                                                                       00500350
SPECS[0]:=SPECS[1]:=SPECS[2]:=SPECS[3]:=0;                                    00500400
NOTE; HADDR.[1:23]:=GT1:=ADDRESS;                                             00500450
IF SCAN AND IDENT THEN                                                        00500500
    BEGIN                                                                     00500600
    TRANSFER(ACCUM,2,SPECS,1,7);                                              00500700
    NOTE;                                                                     00500750
    IF SCAN THEN                                                              00500800
        IF LFTARROW THEN                                                      00500900
            BEGIN                                                             00501000
            SPECS[1]:=1;                                                      00501100
            SPECS[3]:=1;                                                      00501150
            TRANSFER(SPECS,1,SPECS,33,7);                                     00501200
            GT2:=ADDRESS;                                                     00501250
            IF SCAN AND IDENT THEN                                            00501300
                BEGIN                                                         00501400
                TRANSFER(ACCUM,2,SPECS,1,7);                                  00501500
                NOTE;                                                         00501550
                IF SCAN THEN                                          0050160
                    C: IF IDENT THEN                                  0050170
                        BEGIN                                         0050180
                        P:=(SPECS[3]:=SPECS[3]+1)+3;                  0050185
                       TRANSFER(ACCUM,2,SPECS,P8,7);                  0050190
                        SPECS[2]:=1;                                  0050200
                       NOTE;                                          0050205
                        IF SCAN THEN IF IDENT THEN                    0050210
                        BEGIN SPECS[2]:=2;                            0050220
                        P:=(SPECS[3]:=SPECS[3]+1)+2;                  0050225
                       TRANSFER(SPECS,1,SPECS,P8+8,7);                0050230
                       TRANSFER(SPECS,P8,SPECS,1,7);                  0050240
                       TRANSFER(ACCUM,2,SPECS,P8,7);                  0050250
                                                                      0050255
                        B: NOTE; IF SCAN THEN                         0050260
                        A: IF SEMICOLON THEN IF SCAN THEN             0050261
                        IF IDENT THEN                                 0050262
                        BEGIN                                         0050263
                        P:=(SPECS[3]:=SPECS[3]+1)+3;                  0050264
                       TRANSFER(ACCUM,2,SPECS,P8,7);                  0050265
                        GO TO B;                                      0050266
                        END ELSE GO TO A                              0050267
                        ELSE ELSE ELSE                                0050268
                        END ELSE GO TO A                              0050269
                        ELSE END                                      0050270
                    ELSE GO TO A ELSE                                 0050280
            END ELSE ERRORMESS(ERR:=1,GT2,0)                          0050290
            END ELSE GO TO C                                          0050300
        ELSE                                                          0050310
    END ELSE ERRORMESS(ERR:=SYNTAXERROR,GT1,0);                       0050320
FUNCTIONHEADER:=ERR=0;                                                0050450
ADDRESS:=HADDR.[24:24];                                               0050455
END FUNCTIONHEADER;                                                   0050460
                                                                      0080181
INTEGER PROCEDURE DAYTIME(B); ARRAY B[0]; FORWARD;                    0208000
    COMMENT ON LINE 8014000, ARRAY B MUST HAVE LENGTH                 0208001
    AT LEAST 3 WDS;                                                   0208002
PROCEDURE EDITLINE; FORWARD;                                          0208003
INTEGER PROCEDURE LENGTH(A,M);VALUE M; BOOLEAN M; ARRAY A[0];         0208004
    FORWARD; COMMENT LINE 8007900;                                    0208005
BOOLEAN PROCEDURE LABELSCAN(L,K); VALUE K; INTEGER K;                 0208006
    ARRAY L[0]; FORWARD; COMMENT LINE 8013910;                        0208007
                                                                      0208008
                                                                      0208009
PROCEDURE CHECKSEQ(SEQ,L,INC); REAL SEQ,L,INC; FORWARD;               0208010
    COMMENT ON LINE 8040000;                                          0208020
PROCEDURE RELEASEARRAY(D);VALUE D; REAL D;                            0300050
    BEGIN COMMENT  RELEASE PERMANENT STORAGE FOR THE ARRAY DESC D;    0300051
    INTEGER K,J,PT;                                                   0300052
    ARRAY BLOCK[0:32];  %SEE MAXWORDSTORE, LINE 17260                 0300053
    ARRAY TEMP[0:1];                                                  0300053
    IF D.RF NEQ 0 THEN                                                0300054
        BEGIN DELETE1(WS,D.DIMPTR);                                   0300055
        K:=CONTENTS(WS,D.INPTR,BLOCK)-1;                              0300056
        DELETE1(WS,D.INPTR);                                          0300057
        FOR J:=0 STEP 2 UNTIL K DO                                    0300058
        BEGIN TRANSFER(BLOCK,J,TEMP,6,2);                             0300058
            PT:=TEMP[0]; DELETE1(WS,PT); END;                         0300059
        END;                                                          0300060
    END;                                                              0300061
PROCEDURE TRANSFERSP(DIR,SP,L,B,M,N); VALUE DIR,N,M,L;                0300100
    INTEGER DIR,N,M,L;                                                0300110
    ARRAY SP[0,0],B[0];                                               0300120
    BEGIN COMMENT                                                     0300130
    DIR= INTO:    TRANSFER N WORDS FROM B[L] INTO SP[M]               0300140
                  (ACTUALLY SP[*,M] SINCE ARRAY ROW IS USUALLY THE ARG)  0300145
    DIR= OUTOF (OPPOSITE);                                            0300150
    STREAM PROCEDURE MOVER(DIR,SP,M,B,L,N); VALUE DIR,                0300160
        L,M,N;                                                        0300170
        BEGIN LOCAL T;                                                0300180
        SI:=LOC L; DI:=LOC T; DI:=DI+1; DS:=7CHR;                     0300190
        SI:=SP; T(16(SI:=SI+32)); L(SI:=SI+8); L:=SI;                 0300200
        SI:=LOC M; DI:=LOC T; DI:=DI+1: DS:=7CHR;                     0300210
        SI:=B; T(16(SI:=SI+32)); M(SI:=SI+8); M:=SI;                  0300211
        SI:=LOC N; DI:=LOC T; DI:=DI+1; DS:=7CHR;                     0300212
        SI:=LOC DIR; SI:=SI+7;                                        0300213
        IF SC="0" THEN                                                0300214
              BEGIN SI:=M; DI:=L                                      0300215
              END ELSE                                                0300216
              BEGIN SI:=L ; DI:=M                                     0300217
              END;                                                    0300218
        T(2(DS:=32WDS)); DS:=N WDS;                                   0300219
        END;                                                          0300220
```

25

```
     INTEGER K;
     WHILE N:=N-K GTR 0 DO
          MOVER(DIR,SP[(L:=L+K)DIV SPRSIZE,*],
              M:=M+K,B,K:=L MOD SPRSIZE,
              K:=MIN(SPRSIZE-K,N))
     END;

 PROCEDURE DUMPOLISH(SP,PD); VALUE PD; REAL PD; ARRAY SP[0,0];
     BEGIN INTEGER L;
     LABEL SKIPREST;
     INTEGER I,N,M,U; REAL T;
     L:=PD.SPF;
     I:=SP[LOC]+L;
     FOR L:=L+2 STEP 1 UNTIL I DO
 IF (T:=SP[LOC]).TYPEFIELD=FUNCTION THEN
     BEGIN % OUTPUT MESSAGE AND NAME
     FORMWD(2,"5FUNC: ");
     N:=T.LOCFIELD; % N HAS LOCATION OF DESCRIPTOR
     N:=N-1; % BACK UP ONE TO GET NAME
     GTA[0]:=SP[NOC];
     FORMROW(1,1,GTA,1,7);
     END
 ELSE % MIGHT BE AN OPERATOR
 IF T.TYPEFIELD=OPERATOR THEN
          BEGIN COMMENT OUTPUT MESSAGE AND OP CODE;
          FORMWD(2,"5ATOR: ");
          NUMBERCON(T.OPTYPE,ACCUM);
          FORMROW(0,1,ACCUM,2,ACOUNT);
          NUMBERCON(T.LOCFIELD,ACCUM);
          FORMROW(1,1,ACCUM,2,ACOUNT);
          END ELSE %MAY BE A CONSTANT
     IF T.TYPEFIELD=CONSTANT THEN
          BEGIN COMMENT GET DATA DESCRIPTOR;
          N:=T.LOCFIELD;
          FORMWD(2,"5CONS: ");
          T:=SP[NOC];        %T HAS THE DATA DESCRIPTOR
          IF T.SPF=0 THEN BEGIN % A NULL VECTOR
          FORMWD(1,"4NULL   ");
          GO TO SKIPREST; END;
          N:=T.SPF;          %N HAS THE SCALAR OR TOP OF VECTOR LOC.
          IF BOOLEAN(T.SCALAR) THEN M:=U:=N ELSE
               BEGIN U:=SP[NOC]+N; M:=N+1; %UPPER AND LOWER BOUNDS
               END;
 IF BOOLEAN(T.CHRMODE) THEN %CHARACTER FORMAT
     BEGIN COMMENT SP[NOC] IS NUMBER OF CHRS;
     TRANSFERSP(OUTOF,SP,M,BUFFER,0,ENTIER(((T:=
          SP[NOC])-1)DIV 8+1));
     FORMROW(1,1,BUFFER,0,T);
     END ELSE % SHOULD TEST FOR NULL...DO IT LATER.
          FOR N:=M STEP 1 UNTIL U DO
               BEGIN NUMBERCON(SP[NOC],ACCUM);
               FORMROW(0,1,ACCUM,2,ACOUNT);
               END;
          TERPRINT;
          SKIPREST:
          END ELSE COMMENT MUST BE AN OPERAND;
     IF T.TYPEFIELD=LOCALVAR THEN
          BEGIN FORMWD(2,"5LOCL: ");
          N:=T.SPF; % N HAS LOCATION OF NAME;
          GTA[0]:=SP[NOC];   % PUT NAME IN GTA
          FORMROW(1,1,GTA,1,7);
          END ELSE
          BEGIN COMMENT TREAT IT AS VARIABLE;
          N:=T.LOCFIELD; COMMENT N HAS LOC OF DESCRIPTOR;
          N:=N-1; COMMENT BACK UP OVER THE DESCRIPTOR;
          GTA[0]:=SP[NOC];
          FORMWD(2,"5AND  : ");
          FORMROW(1,1,GTA,1,7);
          END;
     END;

PROCEDURE PROCESS(MODE); VALUE MODE; INTEGER MODE;
     BEGIN
     OWN INTEGER J;
     OWN REAL RESULTD;
     LABEL EXPOVRL,INTOVRL,INDEXL,FLAGL,ZEROL;
     MONITOR EXPOVR,INTOVR,INDEX,FLAG,ZERO;
     LABEL DEBUGSP; %DEBUGGING PURPOSES ONLY.
     INTEGER PROCEDURE BUILDCONSTANT(LASTCONSTANT);
          INTEGER LASTCONSTANT; FORWARD;
     INTEGER PROCEDURE GETSPACE(LENGTH); VALUE LENGTH;
          INTEGER LENGTH; FORWARD;
     PROCEDURE OPERANDTOSYMTAB(L);VALUE L;INTEGER L;FORWARD;
```

```
    REAL PROCEDURE BUILDALPHA(LASTCONSTANT);
        INTEGER LASTCONSTANT; FORWARD;
 INTEGER PROCEDURE BUILDNULL(LASTCONSTANT);
    INTEGER LASTCONSTANT; FORWARD;
        PROCEDURE SCRATCHDATA(D);VALUE D;REAL D; FORWARD;
            COMMENT LINE 3121400;
 PROCEDURE FORGETPROGRAM(U);VALUE U;REAL U; FORWARD;
    COMMENT ANALYZE IS IN PROCESS BECAUSE OWN ARRAY SP
       IS ADDRESSED INCORRECTLY OTHERWISE;
 REAL PROCEDURE ANALYZE(DISPLAYOP); VALUE DISPLAYOP;BOOLEAN DISPLAYOP;
    BEGIN COMMENT
    BC= BUILDCONSTANT,
    GS= GET SPACE PROCEDURE ;
    ARRAY INFIX[0:MAXPOLISH];

    INTEGER LASTCONSTANT;
    DEFINE GS=GETSPACE#;
    BOOLEAN STREAM PROCEDURE EQUAL(A,B);
        BEGIN SI:=A; SI:=SI+1; DI:=B; DI:=DI+2;
        IF 7SC=DC THEN TALLY:=1;
        EQUAL:=TALLY;
        END;
 PROCEDURE UNSTACK(DEST,L,ORIG,OTOP,N,CHR1,CHR2);
    VALUE N,CHR1,CHR2;
   INTEGER N,CHR1,CHR2,L,OTOP;
        ARRAY DEST[0,0],ORIG[0];
        BEGIN
        REAL T,U;
        WHILE OTOP GTR 0 AND N GTR 0 AND ERR=0 DO
        IF(IF (T:=ORIG[OTOP]).TYPEFIELD=FUNCTION THEN FALSE ELSE
            U:=T.LOCFIELD=CHR1 OR U=CHR2) THEN %UNSTACK
            BEGIN
            IF N GTR 1  THEN
                IF U=CHR2 THEN ERR:=SYNTAXERROR ELSE
                    OTOP:=OTOP-1;
                N:=N-1;
            END ELSE
           COMMENT WE ARE LOOKING AT AN OPERATOR OR A FUNCTION;


            BEGIN
            IF J NEQ 0 THEN
                BEGIN L:=L+1;
                DEST[LOC]:=ORIG[OTOP]
                END;
            OTOP:=OTOP-1
            END;
      IF N GTR 1 THEN ERR:=SYNTAXERROR;
        END;
    INTEGER ITOP,K,L,I;
    INTEGER M,N,FLOC; REAL T;
    LABEL SKIPSCAN,FILLER;
    LABEL SPFULLAB;


    PROCEDURE FORGETSPACE(L,LENGTH,SP); VALUE L,LENGTH;
        INTEGER L,LENGTH; ARRAY SP[0,0];
        BEGIN IF LENGTH GTR 0 THEN
        BEGIN SP[LOC]:=SP[0,0];
        SP[LOC].LEN:=LENGTH; SP[0,0]:=L
        END;
        END;

    IF CURRENTMODE=FUNCMODE OR STACKBASE=0 THEN FLOC:=0 ELSE

        BEGIN L:=STACKBASE+1;L:=SP[LOC].SPF+1;M:=SP[LOC].SPF+L;
        FLOC:= IF M=L OR BOOLEAN(T:=SP[MOC]).SUSPENDED THEN 0 ELSE T.SPF

        END;

    T:=ADDRESS;
    ITOP:=0;
    DO
    SKIPSCAN:
    IF ITOP LSS MAXPOLISH THEN
        BEGIN
        INFIX[ITOP:=ITOP+1].ADDRFIELD:=T;
        IF SPECIAL THEN
    IF QUOTEV THEN % CONSTANT VECTOR
        BEGIN INFIX[ITOP].TYPEFIELD:=CONSTANT;
        IF T:=BUILDALPHA(LASTCONSTANT) NEQ 0 THEN
        INFIX[ITOP].LOCFIELD:=T ELSE ERR:=SYNTAXERROR
        END ELSE % ORDINARY OPERATOR
```

```
            BEGIN INFIX[ITOP].TYPEFIELD:=OPERATOR;
            INFIX[ITOP].LOCFIELD:=ENTIER(ACCUM[0]);
            END ELSE
        IF NUMERIC THEN
        IF ERR NEQ 0 THEN COMMENT NOTHING; ELSE
            BEGIN INFIX[ITOP].TYPEFIELD:=CONSTANT;
            IF CURRENTMODE=FUNCMODE THEN
            COMMENT DO NOT STORE NUMERIC IN SCRATCH PAD;
            DO UNTIL NOT SCAN OR NOT NUMERIC %THE NULL STATEMENT
            ELSE
                 BEGIN
            T:=BUILDCONSTANT(LASTCONSTANT);
            IF T=0 THEN ERR:=IF ERR=0 THEN VALUEERROR ELSE ERR ELSE
            INFIX[ITOP].LOCFIELD:=T;
                 END;
                IF EOB=0 AND ERR=0 THEN GO TO SKIPSCAN;
            END ELSE
        IF IDENT THEN
                BEGIN INFIX[ITOP].DID:=OPERAND; %SET OPTYPE=NILADIC
            IF NOT(FUNCMODE EQL CURRENTMODE) THEN
                BEGIN J:=0;
                IF FLOC GTR 0 THEN %CHECK LOCAL NAMES
                    BEGIN L:=FLOC+2;
                        K:=SP[LOC]-2;%LAST ALPHA POINTER IN TABLE
                        %SHOULD CONVERT TO BINARY SEARCH
                        T:=L+4;
                        FOR L:=T STEP 2 UNTIL K DO
                            IF EQUAL(SP[LOC],ACCUM) THEN
                            BEGIN J:=L;L:=K;I:=0;
                            INFIX[ITOP].SPF:=J;
                            INFIX[ITOP].RF:=M-FLOC;
                            J:=(J-T+2)/2;
                            END;
                    END;


                IF J EQL 0 THEN
                    BEGIN COMMENT LOOK IN SP SYMBOL TABLE;
                    IF L:=SYMBASE NEQ 0 THEN COMMENT OK TO LOOK;
                        BEGIN T:=SP[LOC];K:=L+T;
                        COMMENT T=N VARS TIMES 2. K IS TOP LIMIT;
                        FOR L:=L +1 STEP 2 UNTIL K DO
                        IF EQUAL(SP[LOC],ACCUM) THEN
                        BEGIN
                        INFIX[ITOP].TYPEFIELD:=I:=SP[LOC].TYPEFIELD;
                        L:=J:=L+1;
                        IF I=FUNCTION THEN BEGIN
                        INFIX[ITOP].RF:=SP[LOC].RETURNVALUE;
                        INFIX[ITOP].OPTYPE:=SP[LOC].NUMBERARGS;END;
                        L:=K;
                        END;
                        IF J EQL 0 THEN
                            IF T LSS MAXSYMBOL×2 THEN %INSERT ID
                            BEGIN L:=K+1; %NEXT AVAILABLE.
FILLER:                     SETFIELD(GTA,0,1,0);
                            TRANSFER(ACCUM,2,GTA,1,7);
                            SP[LOC]:=GTA[0];%STORE VARIABLE NAME
                            OPERANDTOSYMTAB(L);%SET TYPEFIELD AND DESC.
                            IF GT1=FUNCTION THEN%FUNCTION-FIX INFIX
                                BEGIN
                                INFIX[ITOP].OPTYPE:=GTA[1].NUMBERARGS;
                                INFIX[ITOP].TYPEFIELD:=FUNCTION;
                                INFIX[ITOP].RF:=GTA[1].RETURNVALUE;
                                END;
                            J:=L+1;
                            L:=SYMBASE;SP[LOC]:=T+2;%UPDATE SYM TAB #
                            END ELSE SPFULLAB: ERR:=SPERROR;%TAB FULL
                        END ELSE %CREATE SYMBOL TABLE
                        BEGIN
                        SYMBASE:=L:=GS(MAXSYMBOL×2+1);
                        IF ERR NEQ 0 THEN
                            BEGIN SYMBASE:=0;
                            GO TO SPFULLAB;
                          END;
                        T:=0; L:=L+1;
                        GO  TO FILLER;
                        END
                    END ELSE INFIX[ITOP].DID:=LOCALVAR&1[44:47:1];
               INFIX[ITOP].LOCFIELD:=J
                END
              END ELSE ERR:=SYSTEMERROR;
        IF ERR EQL 0 THEN T:=ADDRESS
        END ELSE ERR:=SPERROR
```

28

```
    UNTIL NOT(SCAN AND ERR=0); %DROP THRU WHEN INPUT FIN OR ERR
    COMMENT NOW LOOK FOR THE POLISH;
    IF ERR NEQ 0 THEN
        BEGIN ERRORMESS(ERR,INFIX[ITOP].ADDRFIELD,0);
        END ELSE
        BEGIN COMMENT MAKE UP THE POLISH;
        ARRAY OPERATORS[0:ITOP];
        BOOLEAN PROCEDURE ANDORATOR (VAR,TYPE);
            VALUE VAR, TYPE;
            REAL VAR,TYPE;
            BEGIN
            REAL T;
            LABEL OPERAN, ATOR;
            COMMENT PROCEDURE TRUE IF VAR IS OF TYPE SPECIFIED;
            IF T:=VAR.TYPEFIELD=OPERATOR THEN
                IF T:=VAR.LOCFIELD NEQ RGTPARENV AND T NEQ
                    QQUAD AND T NEQ QUAD AND T NEQ
                    RGTBRACKETV THEN GO ATOR
                ELSE GO OPERAN
            ELSE
            IF T=FUNCTION THEN
                IF VAR.OPTYPE GTR NILADIC THEN
            ATOR:   ANDORATOR:=TYPE=OPERATOR
                ELSE GO OPERAN
            ELSE
            OPERAN: ANDORATOR:=TYPE=OPERAND;
            END OF ANDORATOR;
    BOOLEAN PROCEDURE RGTOPERAND(VAR); VALUE VAR; REAL VAR;
        BEGIN REAL T; DEFINE RT=RGTOPERAND:=TRUE#;
    IF T:=VAR.TYPEFIELD=OPERAND OR T=CONSTANT OR T=LOCALVAR THEN RT
        ELSE IF T=OPERATOR AND VAR.LOCFIELD=LFTPARENV THEN RT
        ELSE IF T=FUNCTION AND VAR.OPTYPE LEQ MONADIC THEN RT;
        END OF RGTOPERAND;
        BOOLEAN VALID;
        INTEGER OTOP;
      INTEGER BCT,N; REAL COLONCTR;
        LABEL STACKOPERAND, STACKFUNCTION;
        DEFINE PTOP=L#;
        LABEL AROUND, NOK, OK, LFTARROWL, LFTPARENL, RGTPARENL,
            SLASHL,EXPL,ROTL,MONADICL,DYADICL,ERRL,SORTL,
            SEMICOLONL, QUADL, DOTL, RELATIONL,
            LFTBRACKETL, RGTBRACKETL, QUOTEQUADL;
        SWITCH OPERATORSWITCH:= %  IN GROUPS OF 5, STARTING AT 1
                NOK, NOK, NOK, LFTARROWL, % 1-4
            MONADICL, SLASHL, OK, LFTPARENL,RGTPARENL, %5-9
            QUADL,LFTBRACKETL,RGTBRACKETL,ERRL,QUOTEQUADL, %10-14
            SEMICOLONL, OK, DOTL, OK, OK, % 15-19
            OK,DYADICL,DYADICL,MONADICL,RELATIONL, % 20-24
                RELATIONL, RELATIONL, RELATIONL, RELATIONL,
                RELATIONL,                % 25-29
            OK, OK, OK, OK, OK, % 30-34
            OK, OK, ROTL, EXPL, OK, % 35-39
            OK,OK,OK,OK,DYADICL, % 40-44
            OK, OK, ERRL, OK, OK, % 45-49
            OK, NOK, NOK, NOK, OK, % 50-54
            SORTL,SORTL,OK,OK,OK, % 55-59
            DYADICL, DYADICL, MONADICL;       % 60-62
        %-----------------------------------------------
    COMMENT GET AN AREA OF SCRATCH PAD IF WE ARE NOT IN
    THE SYNTAX CHECKING MODE;
    J:=(IF CURRENTMODE=FUNCMODE THEN 0 ELSE
    GS(ITOP+3));
        I:=ITOP+1;
    COMMENT A QUICK SYNTAX CHECK;
    IF ANDORATOR(INFIX[ITOP],OPERATOR) THEN ERR:=SYNTAXERROR;
    L:=J+1; COMMENT POLISH WILL START TWO UP IN ARRAY;
    WHILE ERR=0 AND I GTR 1 DO
    IF T:=INFIX[I:=I-1].TYPEFIELD=OPERATOR THEN
    BEGIN
    GO OPERATORSWITCH[INFIX[I].LOCFIELD];
RDTL:
    IF I=1 OR NOT ANDORATOR(INFIX[I-1],OPERAND) THEN GO OK;
    T:=INFIX[I];
    T.LOCFIELD:=ROTATE;
    T.OPTYPE:=IF INFIX[I].OPTYPE NEQ DYADIC THEN MONADIC ELSE DYADIC;
    INFIX[I]:=T; GO TO STACKFUNCTION;
EXPL:
SLASHL:      BEGIN DEFINE STARTSEGMENT= #; %//////////////////////
    IF INFIX[I-1].TYPEFIELD=FUNCTION THEN GO ERRL ELSE
    IF ANDORATOR(INFIX[I-1],OPERATOR) THEN
        BEGIN
        INFIX[I].LOCFIELD:=IF INFIX[I].LOCFIELD=SLASHV THEN
            REDUCT ELSE SCANV;
```

```
          IF INFIX[I].OPTYPE NEQ DYADIC THEN INFIX[I].OPTYPE:=MONADIC;
          GO OK;
          END
     ELSE

     IF INFIX[I].OPTYPE NEQ DYADIC THEN INFIX[I].OPTYPE:=MONADIC;
     IF I=1 THEN

          BEGIN
          ERR:=SYNTAXERROR;
          GO AROUND;
          END;
     GO OK; END;
 SORTL:
     IF I=1 OR ANDORATOR(INFIX[I-1],OPERATOR) THEN GO OK ELSE GO ERRL;
 LFTPARENL:
     K:=I;
     UNSTACK(SP,PTOP,OPERATORS,OTOP,2,RGTPARENV,RGTBRACKETV);
     GO AROUND;
 RELATIONL:
 DYADICL:
     IF I GTR 1 THEN
          IF ANDORATOR(INFIX[I-1],OPERAND) THEN
               BEGIN
               INFIX[I].OPTYPE:=DYADIC;
               GO STACKFUNCTION;
               END;
     IF (GT3:=(T:=INFIX[I+1]).LOCFIELD=REDUCT OR GT3=SCANV)
          AND T.TYPEFIELD=OPERATOR THEN GO OK;
     IF(T:=INFIX[I-1]).LOCFIELD=DOTV AND T.TYPEFIELD=OPERATOR THEN GO OK;
     GO TO ERRL;
 MONADICL:
     IF I=1 OR ANDORATOR(INFIX[I-1],OPERATOR)
          THEN BEGIN
          INFIX[I].OPTYPE:=MONADIC;
          GO TO STACKFUNCTION;
          END
          ELSE
          GO ERRL;
LFTBRACKETL:
     IF BCT:=BCT-1 LSS 0 THEN ERR:=SYNTAXERROR;
    UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTBRACKETV,RGTPARENV);
     IF OTOP=1 THEN BEGIN
          ERR:=SYNTAXERROR; GO AROUND; END
     ELSE IF J NEQ 0 THEN
          BEGIN
          IF T:=INFIX[I-1].TYPEFIELD=OPERAND OR T=LOCALVAR  THEN
       BEGIN DEFINE STARTSEGMENT= #; %/////////////////////////////
           %LFTBRACKET PART OF SUBSCRIPTED VARIABLE
               IF OPERATORS[OTOP].OPTYPE=0 THEN GO TO ERRL;
               COMMENT IF ABOVE TRUE THEN THERE WAS AN OPERAND TO THE RITE;
               L:=L+1;
               N:=GT1:=GETSPACE(1);
               SP[NOC]:=COLONCTR+1; % STORE NUMBER OF DIMENSIONS
               N:=GETSPACE(1); % BUILD A DESCRIPTOR FOR # OF DIMENSIONS
               T.SPF:=GT1;
                T.DID:=DDPNSW;
               T.BACKP:=LASTCONSTANT;
               SP[NOC]:=T;
               T:=INFIX[I];
              T.LOCFIELD:=LASTCONSTANT:=N; % LINK TO CONSTANT CHAIN
              T.TYPEFIELD:=CONSTANT;
              SP[LOC]:=T; % PUT ON POLISH
              L:=L+1;
               IF OPERATORS[OTOP].OPTYPE=3 THEN % LEFT SIDE OF REPLACEOP
               INFIX[I-1].TYPEFIELD:=REPLACELOC;
               SP[LOC]:=INFIX[I-1]; % PLACE OPERAND ON POLISH
               L:=L+1;
               SP[LOC]:=INFIX[I]; % COLLAPSE OPERATOR TO POLISH
               I:=I-1;
               END
        ELSE IF T:=INFIX[I-1].LOCFIELD=SLASHV OR
               T=EXPANDV OR T=ROTV OR T=SORTUPV OR T=SORTDNV THEN
               IF INFIX[I-1].TYPEFIELD=OPERATOR AND OPERATORS[OTOP]
                    .OPTYPE=0 THEN INFIX[I-1].OPTYPE:=DYADIC
              ELSE ERR:=SYNTAXERROR
              ELSE ERR:=SYNTAXERROR;
      END;
        COLONCTR:=OPERATORS[OTOP:=OTOP-1];
        IF OTOP:=OTOP-1 LSS 0 THEN ERR:=SYNTAXERROR;
        GO AROUND;
RGTPARENL:
    IF OTOP LSS ITOP DIV 2 THEN ELSE ERR:=SYNTAXERROR;                03105087
    OPERATORS[OTOP:=OTOP+1]:=INFIX[I];                                03105090
    GO AROUND;                                                        03105100
RGTBRACKETL:  BEGIN DEFINE STARTSEGMENT= #; %////////////////////     03105115
   BCT:=BCT+1;                                                        03105130
    IF OTOP+2 GEQ ITOP THEN                                           03105132
    BEGIN                                                             03105134
    ERR:=SYNTAXERROR;                                                 03105136
    GO AROUND;                                                        03105138
    END;                                                              03105140
   OPERATORS[OTOP:=OTOP+1]:=COLONCTR;                                 03105145
    GT1:=OPERATORS[OTOP:=OTOP+1]:=INFIX[I]; COLONCTR:=0;              03105150
    IF I NEQ ITOP THEN                                                03105152
        IF GT1.OPTYPE NEQ 3 THEN                                      03105154
            OPERATORS[OTOP].OPTYPE:=IF RGTOPERAND(INFIX[I+1]) THEN    03105156
                O ELSE 2                                              03105158
        ELSE                                                          03105159
    ELSE OPERATORS[OTOP].OPTYPE:=2;                                   03105160
        IF J NEQ O AND INFIX[I-1].LOCFIELD=SEMICOLONV THEN            03105161
         BEGIN                                                        03105163
        T.LOCFIELD:=BUILDNULL(LASTCONSTANT);                          03105165
        T.TYPEFIELD:=CONSTANT;                                        03105167
        L:=L+1; K:=I;                                                 03105169
        SP[LOC]:=T;                                                   03105171
        END;                                                          03105173
    GO AROUND; END;                                                   03105175
LFTARROWL:                                                            03105178
    IF I=1 THEN ERR:=SYNTAXERROR                                      03105180
    ELSE                                                              03105182
    IF T:=INFIX[I-1].TYPEFIELD=OPERAND OR T=LOCALVAR THEN             03105184
            INFIX[I-1].TYPEFIELD:=REPLACELOC                          03105186
    ELSE                                                              03105188
    IF T=OPERATOR THEN                                                03105190
        IF T:=INFIX[I-1].LOCFIELD=QUAD OR T=QUADLFTARROW THEN         03105192
            INFIX[I:=I-1].LOCFIELD:=QUADLFTARROW                      03105194
        ELSE IF T=RGTBRACKETV THEN INFIX[I-1].OPTYPE:=3               03105195
        %WILL TEST LATER TO INDICATE REPLACEMENT IN MATRIX 3105154    03105196
            ELSE ERR:=SYNTAXERROR                                     03105197
    ELSE ERR:=SYNTAXERROR;                                            03105198
    IF ERR=0 THEN GO OK ELSE GO AROUND;                               03105200
QUOTEQUADL:                                                           03105202
QUADL:                                                                03105204
    COMMENT INPUT IS BEING REQUESTED;                                 03105205
    GO TO STACKOPERAND;                                               03105206
DOTL:  BEGIN DEFINE STARTSEGMENT=#; %/////////////////////////////    03105207
    IF I GTR 2 THEN                                                   03105208
        IF (T:=INFIX[I-1]).TYPEFIELD=OPERATOR AND                     03105209
            ANDORATOR(T,OPERATOR) THEN                                03105211
            IF (T:=INFIX[I+1]).TYPEFIELD=OPERATOR AND                 03105213
                ANDORATOR(T,OPERATOR) THEN                            03105215
                    IF ANDORATOR(INFIX[I-2],OPERAND) THEN             03105216
            COMMENT THEN SYNTAX OK;                                   03105217
            BEGIN                                                     03105223
            COMMENT STACK OPERATORS SO THAT IF GIVEN A+.XB            03105225
                POLISH IS BA.+X;                                      03105227
            OPERATORS[OTOP].OPTYPE:=TRIADIC;                          03105228
            OPERATORS[OTOP:=OTOP+1]:=INFIX[I-1];                      03105229
            INFIX[I].OPTYPE:=TRIADIC;                                 03105231
            OPERATORS[OTOP:=OTOP+1]:=INFIX[I];                        03105232
            I:=I-1;                                                   03105233
            VALID:=TRUE;                                              03105234
            END;                                                      03105235
        IF NOT VALID THEN ERR:=SYNTAXERROR;                           03105237
        VALID:=FALSE;                                                 03105239
    GO AROUND; END;                                                   03105241
SEMICOLONL:  BEGIN DEFINE STARTSEGMENT=#; %////////////////////       03105242
    IF BCT NEQ 0 THEN                                                 03105244
    BEGIN                                                             03105246
    COLONCTR:=COLONCTR+1;                                             03105248
    IF I-1=0 THEN ERR:=SYNTAXERROR                                    03105250
    ELSE                                                              03105260
    BEGIN                                                             03105262
     UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTBRACKETV,RGTPARENV);         03105265
    IF J NEQ 0 AND (T:=INFIX[I-1].LOCFIELD=SEMICOLONV                 03105270
    OR T =LFTBRACKETV) THEN BEGIN                                     03105280
        T.LOCFIELD:=BUILDNULL(LASTCONSTANT);                          03105290
        T.TYPEFIELD:=CONSTANT;                                        03105300
        L:=L+1; K:=I;                                                 03105310
        SP[LOC]:=T;                                                   03105320
        END;                                                          03105330
    END                                                               03105340
    END                                                               03105350
   ELSE COMMENT MUST BE MIXED MODE EXPRESSION;                        03105370
```

```
          BEGIN
          IF ANDORATOR(T:=INFIX[I-1],OPERATOR) THEN
              IF T.LOCFIELD NEQ SEMICOLONV THEN GO ERRL;
          UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTPARENV,RGTBRACKETV);
          OPERATORS[OTOP:=OTOP+1]:=INFIX[I];
          END;
          GO AROUND;
     END;
NOK:
     ERR:=SYSTEMERROR;
     GO AROUND;
ERRL:
     ERR:=SYNTAXERROR;
     GO AROUND;
OK:
     IF INFIX[I].OPTYPE NEQ 0 THEN GO TO STACKFUNCTION ELSE
     IF I LSS 2 THEN INFIX[I].OPTYPE:=MONADIC ELSE
     INFIX[I].OPTYPE:=IF ANDORATOR(INFIX[I-1],OPERATOR) THEN
          MONADIC ELSE DYADIC;


STACKFUNCTION:
          IF I=K-1 THEN OPERATORS[OTOP:=OTOP+1]:=INFIX[I]
          ELSE
          BEGIN
          UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTPARENV,RGTBRACKETV);
          OPERATORS[OTOP:=OTOP+1]:=INFIX[I];
          END;
          GO AROUND;
AROUND:
     END % OF PROCESSING AN OPERATOR----
          ELSE % COULD BE A FUNCTION
          IF INFIX[I].TYPEFIELD=FUNCTION THEN
               IF (T:=INFIX[I]).OPTYPE GEQ MONADIC THEN
                    GO TO STACKFUNCTION
               ELSE
               IF T.RF=RETURNVAL THEN GO TO STACKOPERAND
               ELSE % MUST NOT RETURN A VALUE
               IF I=1 THEN GO TO STACKOPERAND
               ELSE ERR:=SYNTAXERROR
     ELSE % MUST BE AN OPERAND, CONSTANT OR LOCAL
STACKOPERAND:
          BEGIN DEFINE STARTSEGMENT=#; %///////////////////////////////
          IF ITOP=1 THEN ELSE
          IF I=ITOP AND I NEQ 1 THEN
               IF ANDORATOR(INFIX[I-1],OPERAND) THEN
                    IF INFIX[I-1].LOCFIELD=RGTBRACKETV THEN
                    ELSE GO ERRL
               ELSE
          ELSE
          IF I=1 AND I NEQ ITOP THEN
               IF RGTOPERAND(INFIX[I+1]) THEN GO ERRL
               ELSE
          ELSE
          IF ANDORATOR(INFIX[I-1],OPERAND) OR RGTOPERAND(INFIX[I+1])
               THEN
                    IF INFIX[I-1].LOCFIELD=RGTBRACKETV THEN
                    ELSE GO ERRL;
          IF J NEQ 0 THEN
               BEGIN L:=L+1;
               SP[LOC]:=INFIX[I];
               END; K:=I;
          UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTPARENV,RGTBRACKETV);
          END; % OF GOING THROUGH INFIX
          IF ERR NEQ 0 THEN ERRORMESS(ERR,INFIX[I].ADDRFIELD,0) ELSE
          WHILE OTOP GTR 0 AND ERR=0 DO
          BEGIN IF T:=OPERATORS[OTOP].LOCFIELD=RGTPARENV OR
          T=RGTBRACKETV THEN
               IF OPERATORS[OTOP].TYPEFIELD=OPERATOR THEN
          ERRORMESS(ERR:=SYNTAXERROR,OPERATORS[OTOP].ADDRFIELD
               ,0);
          IF J NEQ 0 THEN
               BEGIN L:=L+1;
               SP[LOC]:=OPERATORS[OTOP]
               END; OTOP:=OTOP-1;
          END;
          IF J NEQ 0 AND DISPLAYOP THEN
               IF SP[LOC].TYPEFIELD NEQ OPERATOR OR
                    T:=SP[LOC].LOCFIELD NEQ LFTARROWV
                    AND T NEQ QUADLFTARROW AND T NEQ GOTOV THEN
                    BEGIN COMMENT ADD DISPLAY OPERATOR TO POLISH;
                    L:=L+1;
                    T.TYPEFIELD:=OPERATOR;
```

```
            T.OPTYPE:=MONADIC;
            T.LOCFIELD:=QUADLFTARROW;
            SP[LOC]:=T;
            END;
      IF J NEQ 0 THEN
          IF ERR NEQ 0 THEN FORGETSPACE (J,ITOP+3,SP) ELSE
          COMMENT STORE POLISH AND BUFFER;
      BEGIN COMMENT SAVE LENGTH OF POLISH;
DEFINE STARTSEGMENT=#; %////////////////////////////////////
      T:=L-J; % DELETE ANY EXTRA SPACE ALLOCATED FOR POLISH
      IF T LSS ITOP+2 THEN FORGETSPACE(L+1,2+ITOP-T,SP);
      COMMENT THEN GETSPACE FOR BUFFER;
L:=GS(((K:=LENGTH(BUFFER, CURRENTMODE=
          CALCMODE))-1) DIV 8 +2);
      COMMENT L IS THE ADDRESS OF THE BUFFER;
      SP[LOC]:=K; %NUMBER OF CHARACTERS IN THE BUFFER
      TRANSFERSP(INTO,SP,L+1,BUFFER,0,ENTIER((K+7)DIV 8));
          COMMENT WE HAVE MOVED IN THE BUFFER;
      K:=L; %SAVE THE ADDRESS OF THE BUFFER;
      L:=J+1; % ONE WORD UP INTO THE POLISH
      SP[LOC].SPF:=K; %STORE ADDRESS OF BUFFER
      SP[LOC].RF:=1;  % SET THE RANK TO 1
      SP[LOC].DID:=DDPNVC;
      L:=L-1;             %SET THE LENGTH OF POLISH
      SP[LOC]:=T;          %STORE THE LENGTH OF THE POLISH
      T:=0; T.SPF:=J; T.RF:=1; %SET UP PROG DESC IN T
      T.BACKP:=LASTCONSTANT;
      T.DID:=PDC; ANALYZE:=T;
      COMMENT DEBUG THE POLISH IF NECESSARY;
     IF POLBUG=1 THEN DUMPOLISH(SP,T);
          END;
      %---------------------------------------------------
      END;
END;
PROCEDURE OPERANDTOSYMTAB(L);VALUE L;INTEGER L;
      BEGIN
      INTEGER N;
      TRANSFER(ACCUM,2,GTA,0,7);
      IF(IF VARIABLES=0 THEN FALSE ELSE
          SEARCHORD(VARIABLES,GTA,GT1,7)=0) THEN
          BEGIN
          SP[LOC].TYPEFIELD:=GT1:=GETFIELD(GTA,7,1);
          IF GT1=FUNCTION THEN
              BEGIN
              L:=L+1;SP[LOC]:=GTA[1];
              END ELSE %MUST BE AN OPERAND
              BEGIN
              SP[LOC].TYPEFIELD:=OPERAND;
              L:=L+1;
              IF GT1=0 THEN % THIS IS THE SCALAR CASE
                  BEGIN N:=GETSPACE(1);
                  SP[LOC]:=N&DDPNSW[CDID];
                  SP[NOC]:=GTA[1];
                  END ELSE %IT MUST BE A VECTOR
                  SP[LOC]:=GTA[1];
              END;
          END ELSE % NOT IN THE SYMBOL TABLE
          BEGIN
          SP[LOC].TYPEFIELD:=GT1:=OPERAND;
          L:=L+1; SP[LOC]:=NAMEDNULLV;
          % THE UNDEFINED SYMBOL IS A NULL

          END;
      END; %OF PROCEDURE OPERANDTOSYMTAB
INTEGER PROCEDURE GETSPACE(LENGTH); VALUE LENGTH;
      INTEGER LENGTH;
      BEGIN
      LABEL ENDGETSPACE,SPOVERFLOW;
      MONITOR INDEX;
      INTEGER L,NEXTAREA,LASTAREA,OLDROW,K;
      INTEGER MEMCHECK;
      REAL LINK;
      INDEX:=SPOVERFLOW;
      NEXTAREA:=SP[0,0];
      LASTAREA:=0;
      DO BEGIN COMMENT FIND A LARGE ENOUGH AREA;
         IF MEMCHECK:=MEMCHECK+1 GTR MAXMEMACCESSES THEN %ERR
      BEGIN GETSPACE:=-1@10; ERR:=SPERROR;
          GO TO ENDGETSPACE END;
          IF NEXTAREA =0 THEN COMMENT END OF STORAGE;
              BEGIN
                  IF NROWS:=(OLDROW:=NROWS)+K:=ENTIER(LENGTH/
                      SPRSIZE+1)
```

```
                    GTR MAXSPROWS THEN %OFF THE END OF SP
                    BEGIN COMMENT TAKE EASY WAY OUT FOR NOW;
                    GETSPACE:=-1@10; %CAUSES INVALID INDEX
                NROWS:=OLDROW; ERR:=SPERROR;
                    GO TO ENDGETSPACE
                    END;
                K:=K×SPRSIZE;

                L:=LASTAREA;
                IF OLDROW = -1 THEN COMMENT FIRST ROW OF SP;
                     BEGIN SP[0,0].NEXT:=L:=1; K:=K-1
                     END ELSE
                BEGIN SP[LOC].NEXT:=(OLDROW+1)×SPRSIZE;
                    L:=(OLDROW+1)×SPRSIZE;
                     END;
                SP[LOC].LEN:=K; SP[LOC].NEXT:=0;
                NEXTAREA:=L
                END ELSE L:=NEXTAREA;
            LINK:=SP[LOC];
            K:=LINK.LEN-LENGTH;
            IF K LSS 0 THEN COMMENT NOT ENOUGH ROOM;
                BEGIN L:=LASTAREA:=NEXTAREA;
                NEXTAREA:=LINK.NEXT
                END
            END UNTIL K GEQ 0;
        IF K GTR 0 THEN
            BEGIN L:=L+LENGTH;
            SP[LOC]:=0;
            SP[LOC].LEN:=K; SP[LOC].NEXT:=LINK.NEXT;
            END ELSE L:=LINK.NEXT;
        K:=L; L:=LASTAREA;
        COMMENT ZERO OUT THE STORAGE BEFORE ALLOCATION;
        SP[LOC].NEXT:=K; K:=NEXTAREA+LENGTH-1;
        FOR L:=GETSPACE:=NEXTAREA STEP 1 UNTIL K DO SP[LOC]:=0;
        IF FALSE THEN SPOVERFLOW: BEGIN
            GETSPACE:=-1@10;ERR:=SPERROR END;
        ENDGETSPACE:
        END OF GETSPACE;
    PROCEDURE FORGETSPACE(LOCATE,LENGTH); VALUE LOCATE,LENGTH;
        INTEGER LOCATE,LENGTH;
        BEGIN INTEGER L;
            IF LENGTH GTR 0 THEN BEGIN
        L:=LOCATE;
        SP[LOC]:=SP[0,0];
        SP[LOC].LEN:=LENGTH;
        SP[0,0]:=L;
            END;
        END;
INTEGER PROCEDURE BUILDNULL(LASTCONSTANT);
    INTEGER LASTCONSTANT;
    BEGIN REAL T, N;
    IF NOT CURRENTMODE=FUNCMODE THEN
    BEGIN
    T:=0;
    T.DID:=DDPNVW;
    T.BACKP:=LASTCONSTANT;
    LASTCONSTANT:=BUILDNULL:=N:=GETSPACE(1);
    SP[NOC]:=T;
    END;
    END OF BUILDNULL;

INTEGER PROCEDURE BUILDCONSTANT(LASTCONSTANT);
INTEGER LASTCONSTANT;
    BEGIN ARRAY A[0:MAXCONSTANT];
    INTEGER ATOP,L,K;
    REAL AP;
    DEFINE GS=GETSPACE#;
    DO
        A[ATOP:=ATOP+1]:=ACCUM[0]
    UNTIL NOT SCAN OR NOT NUMERIC OR ATOP = MAXCONSTANT;
    IF MAXCONSTANT=ATOP OR ERR NEQ 0 THEN COMMENT AN ERROR;
        ELSE

        IF ATOP=1 THEN COMMENT SCALAR FOUND;
            BEGIN L:=K:=GS(1);
            SP[LOC]:=A[1];
            BUILDCONSTANT:=L:=GETSPACE(1);
            SP[LOC]:=K&DDPNSW[CDID]&LASTCONSTANT[CLOCF];
            LASTCONSTANT:=L;
            END ELSE COMMENT VECTOR;
            BEGIN L:=K:=GS(ATOP+1);
            TRANSFERSP(INTO,SP,L+1,A,1,ATOP);
            SP[LOC]:=ATOP;
```

```
          BUILDCONSTANT:=L:=GS(1); %VECTOR DESCRIPTOR
          SP[LOC]:=K&1[CRF]&DDPNVW[CDID]&LASTCONSTANT[CLOCF];
          LASTCONSTANT:=L;
          END

  END;
  OWN INTEGER OLDDATA, REALLYERROR;
  INTEGER L,N,M;
   OWN REAL ST,T,U;
  LABEL EXECUTION,PROCESSEXIT;
  DEFINE STLOC=ST.[30:11],ST.[41:7]#,
  STMINUS=(ST-1).[30:11],(ST-1).[41:7]#,
  AREG=SP[STLOC]#,
  BREG=SP[STMINUS]#,
  BACKPT=6:36:12#,
  CI=18:36:12#,
  SPTSP=30:30:18#,
  PROGMKS=0#,
  IMKS=2#,
  FMKS=1#,

  BACKF=[6:12]#,
  CIF=[18:12]#,
  ENDEF=#;
  PROCEDURE PACK(L,OFFSET,N);VALUE L,OFFSET,N;INTEGER L,OFFSET,N;
       FORWARD;
  INTEGER PROCEDURE UNPACK(S,OFFSET,N);VALUE S,OFFSET,N;
       INTEGER S,OFFSET,N; FORWARD;
  PROCEDURE PUSH;
       IF ST LSS STACKSIZE+STACKBASE THEN ST:=ST+1 ELSE
     ERR:=DEPTHERROR;
PROCEDURE POP;
       BEGIN REAL U;
       IF ST GTR STACKBASE THEN
            IF BOOLEAN((U:=AREG).NAMED)OR NOT BOOLEAN(U.PRESENCE)
                 THEN ST:=ST-1 ELSE
            BEGIN COMMENT GET RID OF SP STORAGE FOR THIS VARIABLE;
            IF U.SPF NEQ 0 AND BOOLEAN(U.DATADESC) THEN
            SCRATCHDATA(U);

            ST:=ST-1;
            END
       ELSE ERR:=SYSTEMERROR;
       END;
  REAL PROCEDURE GETARRAY(DESCRIPTOR); VALUE DESCRIPTOR;
       REAL DESCRIPTOR;
       BEGIN
       INTEGER R,I,J,K,L,LL,TOTAL,PT;
       REAL T;
       ARRAY BLOCK[0:BLOCKSIZE],DIMVECTOR[0:32];
       %SEE MAXWORDSTORE, LINE 17260

     T:=DESCRIPTOR;
     IF (R:=DESCRIPTOR.RF=0) THEN T.DIMPTR:=0
      ELSE BEGIN
          I:=CONTENTS(WS,DESCRIPTOR.DIMPTR,DIMVECTOR);
            TOTAL:=1;
            FOR I:=0 STEP 1 UNTIL R-1 DO
                 TOTAL:=TOTAL×DIMVECTOR[I];
            IF DESCRIPTOR.ARRAYTYPE=CHARARRAY THEN
                 TOTAL:=ENTIER((TOTAL+7) DIV 8);
            TOTAL:=TOTAL+R;
            LL:=GETSPACE(TOTAL);
            TRANSFERSP(INTO,SP,LL,DIMVECTOR,0,R);
            L:=LL+R;
            J:=CONTENTS(WS,DESCRIPTOR.INPTR,DIMVECTOR)-1;
           GTA[0]:=0;
            FOR I:=0 STEP 2 UNTIL J DO
                 BEGIN
               TRANSFER(DIMVECTOR,I,GTA,6,2);
               PT:=GTA[0];
             K:=CONTENTS(WS,PT,BLOCK);
                TRANSFERSP(INTO,SP,L,BLOCK,0,
              (K:=ENTIER((K+7)DIV 8)));
                L:=L+K;
                END;
          T.DIMPTR:=LL;
         END;
     T.INPTR:=0;
     T.PRESENCE:=1;
     GETARRAY:=T;
       END;
  INTEGER PROCEDURE FINDSIZE(D);VALUE D; REAL D;
```

35

```
    BEGIN
    INTEGER I,J,M,R;
 J:=1; I:=D.SPF; R:=D.RF+I-1;
IF I NEQ 0 THEN
 FOR M:=I STEP 1 UNTIL R DO J:=J×SP[MOC];
 FINDSIZE:=J;
 END PROCEDURE FINDSIZE;

INTEGER PROCEDURE NUMELEMENTS(D); VALUE D; REAL D;
    BEGIN
    INTEGER I;
    GT1:=I:=FINDSIZE(D);
    IF D.ARRAYTYPE=CHARARRAY THEN
        I:=ENTIER((I+7) DIV 8);
    NUMELEMENTS:=I;
    END;
PROCEDURE SCRATCHDATA(D); VALUE D; REAL D;
    BEGIN
    INTEGER T,R;
    IF BOOLEAN(D.SCALAR) THEN T:=1 ELSE
        IF R:=D.RF = 0 THEN T:=0 ELSE %BONAFIDE VECTOR
            BEGIN T:=NUMELEMENTS(D)+R;

            END;
    IF T NEQ 0 THEN FORGETSPACE(D.SPF,T);
    END;
COMMENT RELEASEARRAY HAS BEEN MOVED OUT OF PROCESS SO THAT IT
    CAN BE CALLED ELSEWHERE;
REAL PROCEDURE MOVEARRAY(SPDESC); VALUE SPDESC;
    REAL SPDESC;
    COMMENT MOVE THE ARRAY FROM SCRATCHPAD TO PERMANENT
    STORAGE AND CONSTRUCT NEW DESCRIPTOR;
    BEGIN
    INTEGER TOTAL,R,J,M,K;
    REAL T;
ARRAY BLOCK[0:BLOCKSIZE],BUFFER[0:32]; %SEE MAXWORDSTORE, LINE 17260
    T:=SPDESC;
    TRANSFERSP(OUTOF,SP,SPDESC.SPF,BUFFER,0,R:=SPDESC.RF);
    T.DIMPTR:=STORESEQ(WS,BUFFER,8×R);
    TOTAL:=NUMELEMENTS(SPDESC);
    M:=SPDESC.SPF+R;
    K:=ENTIER(TOTAL DIV BLOCKSIZE)-1;
    FOR J:=0 STEP 1 UNTIL K DO BEGIN
        TRANSFERSP(OUTOF,SP,M,BLOCK,0,BLOCKSIZE);
        R:=STORESEQ(WS,BLOCK,BLOCKSIZE×8);
        TRANSFER(R,6,BUFFER,J×2,2);
        M:=M+BLOCKSIZE;
        END;
    IF J:=TOTAL-(K:=K+1)×BLOCKSIZE GTR 0 THEN
        BEGIN
        TRANSFERSP(OUTOF,SP,M,BLOCK,0,J); %GET REMAINDER OF MATRIX
        R:=STORESEQ(WS,BLOCK,J×8);
        TRANSFER(R,6,BUFFER,K×2,2);
        K:=K+1;
        END;
    T.INPTR:=STORESEQ(WS,BUFFER,K×2);
    MOVEARRAY:=T;
    END;
PROCEDURE WRITEBACK;
    COMMENT COPY CHANGED VARIABLES INTO PERMANENT STORAGE;
    BEGIN
    INTEGER I,J,K,L,M,NUM;
    REAL T;
    ARRAY NEWDESC[0:1],OLDDESC [0:1];
    L:=SYMBASE;
    NUM:=SP[LOC]-1;
    L:=L-1;
    FOR I:=1 STEP 2 UNTIL NUM DO BEGIN
        L:=L+2;
        IF ((T:=SP[LOC]).TYPEFIELD) NEQ FUNCTION THEN
          IF BOOLEAN(T.CHANGE) THEN BEGIN
           IF VARIABLES=0 THEN

                BEGIN VARIABLES:=NEXTUNIT;
                T:=CURRENTMODE;
                VARSIZE:=1; STOREPSR;
                CURRENTMODE:=T; VARSIZE:=0;
                END;
           M:=L+1;WHILE(T:=SP[MOC]).BACKP NEQ 0 AND T.PRESENCE=1
           AND(GT1:=GT1+1)LSS MAXMEMACCESSES DO M:=T.BACKP;GT1:=0;
           GTA[0]:=SP[LOC];GTA[1]:=T;
           TRANSFER(GTA,1,NEWDESC,0,7);
```

```
                SETFIELD(NEWDESC,7,1, IF BOOLEAN(T.SCALAR)
                THEN SCALARDATA ELSE ARRAYDATA);
                MOVE(NEWDESC,1,OLDDESC); K:=1;
                IF (IF VARSIZE=0 THEN FALSE ELSE
                    K:=SEARCHORD(VARIABLES,NEWDESC,J,7)=0)
                    THEN BEGIN
                    K:=CONTENTS(VARIABLES,J,OLDDESC);
                    DELETE1(VARIABLES,J);
                    IF GETFIELD(OLDDESC,7,1)=ARRAYDATA THEN
                        RELEASEARRAY(OLDDESC[1]);
                END ELSE
                BEGIN VARSIZE:=VARSIZE+1; J:=J+K-1;
                MOVE(OLDDESC,1,NEWDESC);
                END;
                SETFIELD(NEWDESC,7,1,IF BOOLEAN(T.SCALAR)
                THEN SCALARDATA ELSE ARRAYDATA);
                IF BOOLEAN(T.SCALAR) THEN
                    BEGIN M:=T.SPF;
                    NEWDESC[1]:=SP[MOC];
                    END ELSE %A VECTOR
                    BEGIN T.PRESENCE:=0;
                    NEWDESC[1]:=(IF T.RF NEQ 0 THEN
                    MOVEARRAY(T) ELSE T)
                    END;
                STOREORD(VARIABLES,NEWDESC,J);

                END;
            END;
        END;
    PROCEDURE SPCOPY(S,D,N);VALUE S,D,N;INTEGER S,D,N;
        BEGIN
        INTEGER K;
        WHILE (N:=N-K) GTR 0 DO
        TRANSFERSP(INTO,SP,(D:=D+K),SP[(S:=S+K)DIV SPRSIZE,*],
            K:=S MOD SPRSIZE,K:=MIN(N,SPRSIZE-K));
        END;
    INTEGER PROCEDURE CHAIN(D,CHAINLOC); VALUE D,CHAINLOC;
        INTEGER CHAINLOC; REAL D;
        BEGIN
        INTEGER M;
        CHAIN:=M:=GETSPACE(1);
        D.LOCFIELD:=CHAINLOC;
        SP[MOC]:=D;
        END;
    PROCEDURE SCRATCHAIN(L); VALUE L; INTEGER L;
        BEGIN
        REAL R;
        WHILE L NEQ 0 DO BEGIN
            SCRATCHDATA(R:=SP[LOC]);
            FORGETSPACE(L,1);
            IF L=R.LOCFIELD THEN L:=0 ELSE
            L:=R.LOCFIELD;
            END;
        END;
PROCEDURE RESTORELOCALS(FPTR);VALUE FPTR;REAL FPTR;
    BEGIN
    INTEGER L,M,N,I,K,FLOC;
    REAL T;
    M:=FPTR.LOCFIELD;
    L:=FPTR.SPF+2;K:=SP[LOC]-2;%LAST ALPHA POINTER
    T:=L+4;
    FOR I:=T STEP 2 UNTIL K DO % ONCE FOR EACH LOCAL
        BEGIN
        M:=M+1;N:=SP[MOC].SPF; %LOCATION IN SYMBOL TABLE
        T:=SP[NOC];L:=T.BACKP;T.BACKP:=0;T.NAMED:=0;
        SP[MOC]:=T;%COPY OF DESCRIPTOR TO STACK
        IF L=0 THEN
            BEGIN N:=N-1; GTA[0]:=SP[NOC];
            TRANSFER(GTA,1,ACCUM,2,7); OPERANDTOSYMTAB(N);
            END
        ELSE BEGIN SP[NOC]:=SP[LOC];FORGETSPACE(L,1);END;
        END;
    END; % OF PROCEDURE RESTORELOCALS
    OWN INTEGER FUNCLOC,POLLOC,LASTMKS,POLTOP,CINDEX;
    PROCEDURE STEPLINE(LABELED); VALUE LABELED;
        BOOLEAN LABELED;

        BEGIN
        LABEL ENDFUNC,TERMINATE,DONE;
        LABEL BUMPLINE;
        LABEL TRYNEXT;
        REAL STREAM PROCEDURE CON(A); VALUE A;
            BEGIN SI:= LOC A; DI:=LOC CON; DS:=8DEC;
```

```
      INTEGER C;
      REAL N,T,L,TLAST,M,BASE;
          COMMENT
   MONITOR PRINT (FUNCLOC,POLLOC,LASTMKS,POLTOP,CINDEX,N,T,L,
      TLAST,M,BASE);
      L:=FUNCLOC;M:=SP[LOC].SPF+L;
      IF BOOLEAN(SP[MOC],SUSPENDED) THEN
          BEGIN %RESUME A SUSPENDED FUNCTION
          SP[MOC].SUSPENDED:=0;%REMOVE SUSPENDED BIT
          RESTORELOCALS(SP[MOC]);
          SP[LOC].RF:=N:=SP[LOC].RF-1;
          IF N LEQ 0 THEN SUSPENSION:=0;% NO MORE SUSPENDED FNS
          END;
      IF LABELED THEN %MAKE INTIAL CHECKS AND CHANGES;
          BEGIN
          IF NOT BOOLEAN((T:=AREG).PRESENCE) OR L:=T.SPF=0
              THEN
              BEGIN LABELED:=FALSE; GO TO BUMPLINE;
              END;
          IF BOOLEAN (T.CHRMODE) THEN GO TO TERMINATE;
          L:=L+T.RF; %PICK UP THE FIRST ELEMENT OF THE ARRAY
          IF T:=SP[LOC] GTR 9999.99994 OR T LSS 0 THEN
              T:=0;
          T:=CON(ENTIER(T×10000+.5))
          END; BUMPLINE:
      L:=LASTMKS; TLAST:=SP[LOC].BACKF;
      C:=(LASTMKS:=SP[MOC].LOCFIELD)-STACKBASE;%LOC OF FMKS
      WHILE TLAST GTR C DO %STRIP OFF CURRENT LINE
          BEGIN L:=TLAST+STACKBASE;TLAST:=(N:=SP[LOC]).BACKF;
          IF N.DID=IMKS THEN SCRATCHAIN(N.SPF);
          END;
      WHILE ST GEQ L AND ERR=0 DO POP;
      IF ERR NEQ 0 THEN GO TO DONE;
      M:=BASE:=SP[MOC].SPF;%LOC OF LABEL TABLE
 TRYNEXT:
      N:=SP[MOC]+M+1;  % N IS ONE BIGGER THAN TOP
      M:=M+2; M:=SP[MOC]+2;  % M IS ON THE FIRST POINTER
      IF LABELED THEN %BINARY SEARCH FOR THE DESIRED LINE
          BEGIN
          IF N-M LSS 2 THEN GO TO ENDFUNC;
          WHILE N-M GTR 2 AND C LSS 1@8 DO

              BEGIN L:=M+ENTIER((N-M)DIV 4)×2; C:=C+1;
              IF T LSS SP[LOC] THEN N:=L ELSE M:=L
              END;
      IF C=1@8 THEN GO TERMINATE;
          IF SP[MOC] NEQ T THEN GO ENDFUNC; T:=M;
          %T HAS THE SP LOCATION OF THE CORRECT LABEL
          END ELSE %BUMP THE POINTER
      IF T:=CURLINE+2+BASE GEQ N OR T LSS M THEN GO ENDFUNC;
      M:=T+1; CURLINE:=T-BASE; %M IS SET TO PROG DESC
      IF NOT BOOLEAN((T:=SP[MOC]).PRESENCE) THEN %MAKE POLISH
          BEGIN N:=BASE+1;N:=SP[NOC].SPF;%SEQ STORAGE UNIT
          INITBUFF(BUFFER,BUFFSIZE);
          N:=CONTENTS(N,T,BUFFER); %GET TEXT
          RESCANLINE; WHILE LABELSCAN(GTA,0) DO; %CLEAR LABELS
          IF BOOLEAN(EOB) THEN % AN EMPTY LINE--BUMP POINTER
              BEGIN M:=BASE;LABELED:=FALSE;GO TO TRYNEXT;END ELSE
          IF T:=ANALYZE(TRUE)=0 THEN % NO GOOD
              GO TO DONE;
          SP[MOC]:=T; %SAVE THE POLISH DESCRIPTOR AT M
          END ;
      PUSH; IF ERR NEQ 0 THEN GO TO DONE;
      AREG:=(L:=ENTIER(M))&1[CCIF]&TLAST[BACKPT];
      LASTMKS:=ST;
      POLLOC:=SP[LOC].SPF;
      L:=T.SPF; POLTOP:=SP[LOC]; CINDEX:=1;
      GO TO DONE;
ENDFUNC:
      %ARRIVE HERE WHEN FUNCTION IS COMPLETED.
      %GET RESULT OF FUNCTION
      M:=FUNCLOC;M:=SP[MOC].SPF+M;N:=TLAST:=SP[MOC].LOCFIELD;
      M:=SP[NOC].SPF;M:=SP[MOC];
      COMMENT I CANNOT CONJURE UP A CASE WHERE A USER RETURNS TO A
      FUNCTION WHOSE DESCRIPTOR HAS BEEN PUSHED DOWN BY A SUSPENDED
      VARIABLE.IF THIS HAPPENS-HOPE FOR A GRACEFUL CRASH;
      %M IS THE DESCRIPTOR FOR THE FUNCTION, TLAST  IS BASE ADDRESS

      IF BOOLEAN(M.RETURNVALUE) THEN %GET THE RESULT
         BEGIN
         N:=M.SPF+5;%RELATIVE LOCATION OF RESULT
         N:=SP[NOC]+TLAST; %LOCATION IN STACK OF RESLULT
```

```
            T:=SP[NOC]; SP[NOC].NAMED:=1; N:=T;
            END;
        WHILE ST GEQ TLAST AND ERR=0 DO POP; %GET RID OF TEMPS
        OLDDATA:=(T:=AREG).SPF; POP;% GET RID OF INTERRUPT MKS
        IF ERR NEQ 0 THEN GO TO DONE;
        IF BOOLEAN(M.RETURNVALUE) THEN %REPLACE RESULT
            BEGIN PUSH; IF ERR NEQ 0 THEN GO TO DONE;
            AREG:=N; %RESULT OF CALL
            END;
        L:=STACKBASE+1;L:=SP[LOC].SPF+1;M:=SP[LOC].SPF+L;

        SP[MOC]:=0;SP[LOC].SPF:=(M:=M-1)-L;
        COMMENT NOW INITIATE ANY OLD FUNCTIONS, AND GET POLISH
        GOING;
        LASTMKS:=N:=T.BACKF+STACKBASE; %LOCATION OF PROGRAM DESC.
        T:=SP[NOC]; % PICK UP PROGRAM DESCRIPTOR
        N:=T.SPF; %LOCATION OF POLISH DESCRIPTOR
        POLLOC:=(N:=SP[NOC].SPF);
        POLTOP:=SP[NOC];
        CINDEX:=T.CIF;
        IF M NEQ L THEN % GET LAST FUNCTION STARTED
            BEGIN N:=SP[MOC].LOCFIELD;
            T:=SP[NOC];
            CURLINE:=T.CIF
            END ELSE CURLINE:=0;
        GO TO DONE;
TERMINATE:
        ERR:=LABELERROR;
DONE:
        END;

PROCEDURE FIXTAKEORDROP(LDESC,RDESC,OPT,MAP,SIZEMAP,SIZE);
  VALUE LDESC,RDESC,OPT; REAL LDESC,RDESC;
  INTEGER OPT, SIZE; ARRAY MAP, SIZEMAP [1];
  BEGIN INTEGER LRANK,LSIZE,L,M,RRANK,N,I,TOP,PUT;
    DEFINE TAKE = OPT = 2#;
    INTEGER LNUM, RNUM; LABEL QUIT;
  IF LSIZE := FINDSIZE(LDESC) NEQ RRANK := RDESC.RF AND LSIZE NEQ 1
    OR LRANK:=LDESC.RF GTR 1 AND LSIZE NEQ 1
    OR L := LDESC.SPF=0
    OR M := RDESC.SPF = 0 THEN BEGIN
       ERR:=DOMAINERROR; GO TO QUIT; END;
  L := L + LRANK;

  SIZE := 1;
  FOR I := 1 STEP 1 UNTIL RRANK DO BEGIN
    RNUM:=SP[MOC];
    LNUM:=IF TAKE THEN SP[LOC] ELSE (PUT:=SP[LOC])-SIGN(PUT)×RNUM;
    IF ABS(LNUM) GTR RNUM THEN BEGIN
      ERR:=DOMAINERROR; GO TO QUIT; END;
    IF LNUM = 0 THEN BEGIN
       SIZE := 0; GO TO QUIT; END;
    IF LNUM GTR 0 THEN BEGIN
      SIZEMAP[I] := LNUM;
      MAP[I] . SPF := 0;
      MAP[I] . RF := 1;
    END ELSE BEGIN
      LNUM:=ABS(LNUM);
      PUT := RNUM - LNUM + ORIGIN;
     MAP[I].SPF := N := GETSPACE(LNUM+1);
      SIZEMAP[I] := SP[NOC] := LNUM;
      TOP := N + LNUM;
      FOR N:=N+1 STEP 1 UNTIL TOP DO BEGIN
        SP[NOC]:=PUT; PUT:=PUT+1; END;
      MAP[I].RF := 1;
      MAP[I] := - MAP[I];
      END;
    IF LSIZE NEQ 1 THEN L:=L+1;
    M:=M+1;
    SIZE:=SIZE × LNUM;
    END;
 QUIT:  END PROCEDURE FIXTAKEORDROP;
    REAL PROCEDURE SUBSCRIPTS(DIRECTION,D,RANK);
        VALUE DIRECTION,D,RANK; REAL D,RANK; INTEGER DIRECTION;
        BEGIN COMMENT THIS PROCEDURE EVALUATES A SET OF SUBSCRIPTS
        ,POPS THEM OFF OF THE STACK, AND RETURNS WITH A DESC.
        FOR THE ITEM REFERENCED;
        LABEL GOHOME,DONE;
        INTEGER SIZE,I,L,M,N,VALUW;
        INTEGER ADDRESS,NOTSCAL,DIM,LEVEL,TEMP,K,J;
        REAL SUBDESC,T;
        BOOLEAN DCHARS;
        STREAM PROCEDURE TCHAR(A,B,C,D);VALUE B,D;
```

39

```
          BEGIN SI:=A;SI:=SI+B;DI:=C;DI:=DI+D;DS:=CHR;END;
      ARRAY MAP[1:RANK],SIZEMAP[1:RANK];
      ARRAY BLOCKSIZE[1:RANK],POINTER[0:RANK],PROGRESS[1:RANK];
      INTEGER PROCEDURE SUBINDEX(M,S,P);VALUE M,S,P;REAL M,S,P;
          IF M LSS 0 THEN BEGIN M:=-M;
               M:=P+M.SPF+M.RF-1;SUBINDEX:=SP[MOC]-ORIGIN;END
               ELSE SUBINDEX:=(IF S=1 THEN M.SPF ELSE M.SPF+P-1);
      COMMENT
      MONITOR PRINT(I,L,M,N,VALUW,ADDRESS,T,ERR,MAP,SIZEMAP,
       SIZE,D,RANK,DIRECTION);
      DCHARS:=BOOLEAN(D.CHRMODE);
      IF DIRECTION GTR 1 THEN % THIS IS TAKE OR DROP
          BEGIN
          NOTSCAL:=1;
      FIXTAKEORDROP(AREG,BREG,DIRECTION,MAP,SIZEMAP,SIZE);
      IF ERR NEQ 0 THEN GO TO GOHOME;
      IF SIZE=0 THEN BEGIN D.DID:=DDPUVW; D.RF:=1;
        D.SPF:=0; SUBSCRIPTS:=D; GO TO GOHOME; END;
        %IF SIZE=0 AND TAKE OR DROP, RESULT IS A NULL
      END ELSE BEGIN
      IF RANK NEQ D.RF THEN BEGIN ERR:=RANKERROR;GO TO GOHOME;END;
      SIZE:=1;
      N:=D.SPF-1;
      L:=ST-1; % LOCATE THE EXECUTION STACK
      FOR I:=1 STEP 1 UNTIL RANK DO
          BEGIN
          L:=L-1; SUBDESC:=SP[LOC]; % WANDER INTO EXEC STACK
          IF ERR NEQ 0 THEN GO TO GOHOME;
          N:=N+1;
          IF BOOLEAN(SUBDESC.SCALAR) THEN
               BEGIN M:=SUBDESC.SPF;
               IF (VALUW:=SP[MOC]-ORIGIN) GEQ SP[NOC]
               OR VALUW LSS 0 THEN BEGIN ERR:=INDEXERROR;GO TO
               GOHOME; END;
               MAP[I]:=VALUW; SIZEMAP[I]:=1;
               END ELSE % CHECK FOR A NULL
               IF SUBDESC.SPF=0 THEN % THIS IS A NULL
                    BEGIN
                    NOTSCAL:=1;
                    SIZE:=SIZE×(M:=SP[NOC]);
                    MAP[I].RF:=1;SIZEMAP[I]:=M;
                    END ELSE % IT MUST BE A VECTOR
      BEGIN DEFINE STARTSEGMENT=#; %///////////////////////////////


                    NOTSCAL:= 1;
                    MAP[I]:=-((M:=SUBDESC.SPF)&SUBDESC.RF[CRF]);
                    SIZE:=SIZE×(SIZEMAP[I]:=FINDSIZE(SUBDESC));
                    J:=SP[NOC]+ORIGIN;M:=M+SUBDESC.RF;T:=SIZEMAP[I]+M
                    -1;
                    FOR M:=M STEP 1 UNTIL T DO
                         IF SP[MOC] GEQ J OR SP[MOC] LSS ORIGIN THEN
                         BEGIN ERR:=INDEXERROR; GO TO GOHOME; END;
                    END;
          END; % OF THE FOR STATEMENT
      END;
      IF SIZE LEQ 0 THEN BEGIN ERR:=INDEXERROR;GO TO GOHOME;END;
      IF SIZE=1 AND NOT BOOLEAN(NOTSCAL) THEN %SCALAR REFERENCED
          BEGIN
  DEFINE STARTSEGMENT=#; %///////////////////////////////////
          N:=D.SPF; M:=RANK-1;
          FOR I:=1 STEP 1 UNTIL M DO
               BEGIN N:= N+1;
               ADDRESS:=SP[NOC]×(ADDRESS+MAP[I]);
               END;
          ADDRESS:=ADDRESS+MAP[RANK] +1;
          IF DIRECTION=OUTOF THEN
               IF DCHARS THEN BEGIN
                    N:=(ADDRESS+7)DIV 8+N;J:=(ADDRESS-1)MOD 8;
                    T:=M:=GETSPACE(2);SP[MOC]:=1;M:=M+1;
                    SP[MOC]:=0; TCHAR(SP[NOC],J,SP[MOC],0);
                    SUBSCRIPTS:=T&1[CRF]&DDPUVC[CDID];
                    END ELSE
               BEGIN N:= ADDRESS+N;
               M:=GETSPACE(1);SP[MOC]:=SP[NOC];
               T:=M; T.DID:=DDPUSW;
               SUBSCRIPTS:=T;
               END ELSE % DIRECTION IS INTO
               BEGIN
               L:=L-1;SUBSCRIPTS:=SUBDESC:=SP[LOC];
               IF DCHARS AND FINDSIZE(SUBDESC)=1 OR
                    BOOLEAN(SUBDESC.SCALAR) THEN
                    BEGIN
```

```
                L:=GETSPACE(N:=(NUMELEMENTS(D)+D.RF));
                SPCOPY(D.SPF,L,N); % MAKE A NEW COPY
                IF DCHARS THEN BEGIN
                N:=(ADDRESS+7)DIV 8+L;J:=(ADDRESS-1)MOD 8;
                M:=SUBDESC.SPF;IF SP[MOC] GTR 1 OR SUBDESC.RF
                    NEQ 1 THEN BEGIN ERR:=DOMAINERROR;GO TO
                    GOHOME;END;
                M:=M+1;TCHAR(SP[MOC],0,SP[NOC],J);
                END ELSE BEGIN
                M:=L+ADDRESS+D.RF-1;
                N:=SUBDESC.SPF;
                SP[MOC]:=SP[NOC]; %PERFORM THE REPLACEMENT
                END;
                N:=D.LOCFIELD;I:=SP[NOC].BACKP;
                SP[NOC]:=D&L[CSPF]&I[CLOCF];%STORE NEW DESC
                OLDDATA:=CHAIN(D,OLDDATA);
                IF BOOLEAN(D.NAMED) THEN BEGIN
                    N:=N-1;IF I=0 AND SP[NOC].SUSPENDVAR=0
                    THEN SP[NOC].CHANGE:=1%MUST BE A REAL GLOBAL
                    END ELSE %MUST BE A LOCAL VARIABLE
                    AREG.NAMED:=1; %DONT LET IT BE FORGOTTEN
                END ELSE ERR:=RANKERROR;
            END;
        END ELSE % A VECTOR IS REFERENCED
        BEGIN % START WITH INITIALIZATION
        N:=D.SPF+D.RF;BLOCKSIZE[RANK]:=PROGRESS[RANK]:=J:=1;
        FOR I:=RANK-1 STEP -1 UNTIL 1 DO
            BEGIN N:=N-1;
            J:=BLOCKSIZE[I]:=J×SP[NOC];
            PROGRESS[I]:=1;
            END;
        K:=POINTER[1]:=SUBINDEX(MAP[1],SIZEMAP[1],PROGRESS[1])
            ×BLOCKSIZE[1];
        FOR I:=2 STEP 1 UNTIL RANK DO
            K:=POINTER[I]:=K+SUBINDEX(MAP[I],SIZEMAP[1],
                PROGRESS[I])×BLOCKSIZE[I];
        DIM:=0;
        FOR I:=1 STEP 1 UNTIL RANK DO
            IF SIZEMAP[I] GTR 1 THEN DIM:=DIM+MAP[I].RF;
        IF DCHARS THEN BEGIN TEMP:=D; D.SPF:=UNPACK(D.SPF,
            RANK,FINDSIZE(D)); IF DIM=0 THEN DIM:=1; END;
    IF DIRECTION GTR 0 THEN % OUTOF..TAKE.. OR DROP
    BEGIN DEFINE STARTSEGMENT=#; %///////////////////////////
IF SIZE+DIM GTR MAXWORDSTORE THEN BEGIN ERR:=KITEERROR; GO TO
   GOHOME END ELSE TEMP:=L:=GETSPACE(SIZE+DIM); %ROOM FOR RESULT
            IF DIM GTR 0 THEN
            IF DIM=1 THEN BEGIN SP[LOC]:=SIZE; L:=L+1;END
            ELSE FOR I:=1 STEP 1 UNTIL RANK DO
                IF SIZEMAP[I] GTR 1 THEN
                IF (M:=MAP[I].SPF)=0 THEN BEGIN SP[LOC]:=
                    SIZEMAP[I];L:=L+1;END ELSE
                    BEGIN N:=M+MAP[I].RF-1;

                    FOR M:=M STEP 1 UNTIL N DO BEGIN
                    SP[LOC]:=SP[MOC];L:=L+1;END;
                    END;
                COMMENT THIS INITIALIZES RESULT DIM VECTOR;
            ADDRESS:= D.SPF+D.RF;
            END ELSE % DIRECTION IS INTO
             BEGIN DEFINE STARTSEGMENT=#; %////////////////
              L:=L-1; SUBSCRIPTS:=SUBDESC:=SP[LOC];
            IF FINDSIZE(SUBDESC) NEQ SIZE THEN
                BEGIN ERR:=RANKERROR; GO TO GOHOME;END;
            N:=SUBDESC.RF;
            IF BOOLEAN(SUBDESC.CHRMODE) THEN SUBDESC.SPF:=
                UNPACK(SUBDESC.SPF,N,FINDSIZE(SUBDESC));
            IF DCHARS THEN L:= D.SPF ELSE BEGIN
            L:=GETSPACE(N:=(NUMELEMENTS(D)+D.RF));
            SPCOPY(D.SPF,L,N); % MAKE FRESH COPY TO PATCH INTO
            END;
            ADDRESS:=L+D.RF; % SP LOCATION TO STORE INTO
            N:=D.LOCFIELD;I:=SP[NOC].BACKP;
            SP[NOC]:=D&L[CSPF]&I[CLOCF];%STORE NEW DESC.
            OLDDATA:=CHAIN(IF DCHARS THEN TEMP ELSE D,OLDDATA);
            IF BOOLEAN(D.NAMED ) THEN BEGIN
                N:=N-1;IF I=0 AND SP[NOC].SUSPENDVAR=0
                THEN SP[NOC].CHANGE:=1%MUST BE A REAL GLOBAL
                END ELSE %IT MUST BE A LOCAL VARIABLE
                AREG.NAMED:=1;%DONT LET IT BE FORGOTTEN ON POP
            L:=SUBDESC.SPF+SUBDESC.RF;%POINT TO SOURCE
            END;
```

```
         WHILE TRUE DO % RECURSIVE EVALUATION LOOP
            BEGIN N:=POINTER[RANK]+ADDRESS;
            LEVEL:=RANK;
         IF DIRECTION GTR 0 THEN  %OUTOF..TAKE..DROP
               BEGIN SP[LOC]:=SP[NOC]; L:=L+1;
               END ELSE BEGIN % INTO
               SP[NOC]:= SP[LOC];L:=L+1; END;
            WHILE PROGRESS[LEVEL]GEQ SIZEMAP[LEVEL] DO
               BEGIN PROGRESS[LEVEL]:=1 ; %LOOK FOR MORE WORK
               IF LEVEL:=LEVEL-1 LEQ 0 THEN GO TO DONE;
               END;
            COMMENT THERE IS MORE ON THIS LEVEL;
            PROGRESS[LEVEL]:=PROGRESS[LEVEL]+1;
            K:=POINTER[LEVEL]:=POINTER[LEVEL-1] +SUBINDEX(
               MAP[LEVEL],SIZEMAP[LEVEL],PROGRESS[LEVEL])×
               BLOCKSIZE[LEVEL];%POINTER[0] IS 0
            FOR I:=LEVEL+1 STEP 1 UNTIL RANK DO
               K:=POINTER[I]:=K+SUBINDEX(MAP[I],SIZEMAP[I],
               PROGRESS[I])×BLOCKSIZE[I];
            END; % OF RECURSIVE EVALUATION LOOP
   DONE:   IF DIRECTION GTR 0 THEN % OUTOF TAKE OR DROP
            IF DCHARS THEN BEGIN PACK(TEMP,DIM,SIZE);
               FORGETSPACE(D.SPF,RANK+FINDSIZE(D));
               SUBSCRIPTS:=TEMP&DIM[CRF]&DDPUVC[CDID];
               END ELSE % THIS IS A NUMERIC VECTOR
            IF DIM=0 THEN SUBSCRIPTS:=TEMP&DDPUSW[CDID] ELSE
            SUBSCRIPTS:=TEMP&DIM[CRF]&DDPUVW[CDID]
            ELSE % THE DIRECTION IS INTO
            BEGIN IF BOOLEAN(SUBDESC.CHRMODE) THEN
               FORGETSPACE(SUBDESC.SPF,FINDSIZE(SUBDESC)+1);
            IF DCHARS THEN PACK(D.SPF,RANK,FINDSIZE(D));
            END;

         END;
GOHOME:  IF DIRECTION GTR 1 THEN
    FOR I:=1 STEP 1 UNTIL RANK DO
     IF MAP[I] LSS 0 THEN FORGETSPACE(MAP[I].SPF,SIZEMAP[I]+1);
       END; % OF SUBSCRIPTS PROCEDURE
    PROCEDURE TMS(N); VALUE N; INTEGER N;
       BEGIN COMMENT N=0 FOR REGULAR INTERRUPT MKS
                     N=1 FOR QQUAD INTERRUPT MKS
                     N=2 FOR QUAD INTERRUPT MKS
                     N=3 FOR EXECUTION LINE FOLLOWING
                     N=4 FOR SUSPENDED FUNCTION;
       INTEGER L,M;

       PUSH;AREG:=OLDDATA&(LASTMKS-STACKBASE)
          [BACKPT]&N[QUADINV]&IMKS[CDID];
       IF N NEQ 4 THEN BEGIN L:=LASTMKS;SP[LOC].CIF:=CINDEX;END;
       L:=STACKBASE+1;L:=SP[LOC].SPF +1;
       IF (M:=SP[LOC].SPF) NEQ 0 THEN % SAVE CURLINE
          BEGIN L:=L+M; L:=SP[LOC].LOCFIELD;
          SP[LOC].CIF:=CURLINE;
          END;
       LASTMKS:=ST;
       END;
PROCEDURE DISPLAYCHARV(D); VALUE D; REAL D;
    BEGIN INTEGER I,J,K,L,M,NWORDS,NJ,T,NMAT,II,JJ,WDLINE,F,CC;
    COMMENT WDLINE=#WORDS NEEDED TO FILL A TELETYPE LINE
            NWORDS=#WORDS NEEDED TO GET F CHARACTERS FOR LAST
                         TELETYPE LINE OF A ROW
          F=#CHARACTERS IN LAST TELETYPE LINE OF A ROW
          T=#TELETYPE LINES NEEDED PER ROW BEYOND FIRST LINE
          NMAT=#MATRICES TO BE PRINTED OUT (1 IF RANK=2);
    L := (T:=D.SPF) + (NJ:=D.RF) - 1;
    J := SP[LOC];  %J IS NUMBER OF CHARACTERS PER ROW
    IF NJ GTR 1 THEN BEGIN
       L:=L-1; K:=SP[LOC]
       END ELSE K := 1;  %K IS NUMBER OF ROWS PER MATRIX

    L := T + NJ;
       NMAT := FINDSIZE(D) DIV (J×K);
       WDLINE := (LINESIZE+6) DIV 8 + 1;
       IF II:=J-LINESIZE GTR 0 THEN BEGIN
          T:=II DIV (I:=LINESIZE-2)+(IF II MOD I=0 THEN 0 ELSE 1);
          NWORDS:=((F:=II-(T-1)×T)+6) DIV 8 + 1;
       END ELSE BEGIN NWORDS:=((F:=J)+6)DIV 8 + 1; T:=0; END;
       FOR II:=1 STEP 1 UNTIL NMAT DO BEGIN
          FOR I:=1 STEP 1 UNTIL K DO BEGIN
            CC:=0;
            FOR JJ:=1 STEP 1 UNTIL T DO BEGIN
              TRANSFERSP(OUTOF,SP,L+M DIV 8,BUFFER,0,WDLINE);
              FORMROW(3,CC,BUFFER,ENTIER(M MOD 8),NJ:=LINESIZE-CC);
              M := M + NJ;  CC := 2;  END;
          IF I=K AND II=NMAT THEN IF L+M DIV 8 + NWORDS GTR
             (1+NROWS)×SPRSIZE THEN NWORDS:=NWORDS-1;
             %TO TAKE CARE OF BEING AT END OF SP
          TRANSFERSP(OUTOF,SP,L+M DIV 8, BUFFER,0,NWORDS);
          FORMROW(3,CC,BUFFER,ENTIER(M MOD 8), F);
          M := M + F;
          END;
        FORMWD(3,"1        ");
      END;
  END OF CHARACTER DISPLAY PROCEDURE;
 REAL PROCEDURE SEMICOL;
      BEGIN COMMENT FORM CHAR STRING FROM TWO DESCRIPTORS;
      INTEGER J,K,L;
      REAL LD, RD;
      STREAM PROCEDURE BLANKS(B,J,K);VALUE J,K;
          BEGIN LOCAL T,U;
          SI:=LOC K; DI:=LOC U; DI:=DI+1; DS:=7 CHR;
          SI:=LOC J; DI:=LOC T; DI:=DI+1; DS:=7 CHR;
          DI:=B; U(2(DI:=DI+32)); DI:= DI+K;
          T(2(DS:=32 LIT " ")); J(DS:=1 LIT " ");
          END;
      PROCEDURE MOVEC(J,L,K);VALUE J,L,K; INTEGER J,L,K;
          BEGIN INTEGER I;
          IF(J+K+8) GTR MAXBUFFSIZE×8 THEN ERR:=LENGTHERROR ELSE
              BEGIN TRANSFERSP(OUTOF,SP,L,BUFFER,ENTIER((J+7)DIV 8),
                  ENTIER((K+7) DIV 8));
              IF I:=(J MOD 8) NEQ 0 THEN TRANSFER(BUFFER,J+8-I,
              BUFFER,J,K); END;
          END;
       INTEGER PROCEDURE MOVEN(J,L,K);VALUE J,L,K;INTEGER J,L,K;
          BEGIN INTEGER I;K:=K+L-1; I:=MAXBUFFSIZE×8;
          BLANKS(BUFFER,I-J,J);
          FOR L:= L STEP 1 UNTIL K DO
              BEGIN NUMBERCON(SP[LOC],ACCUM);
              TRANSFER(ACCUM,2,BUFFER,J:=J+1,ACOUNT);
              IF (J:=J+ACOUNT)GTR I THEN BEGIN L:=K;ERR:=LENGTHERROR;
              END;END;
          MOVEN:=J;
          END;
     LD := AREG; RD := BREG;
      IF L:=LD.RF GTR 1 THEN ERR:= RANKERROR ELSE
          IF LD.SPF NEQ 0 THEN
          IF BOOLEAN(LD.CHRMODE) THEN MOVEC(0,L+LD.SPF,J:=FINDSIZE
              (LD))ELSE J:=MOVEN(0,L+LD.SPF,FINDSIZE(LD));
          IF L:=RD.RF GTR 1 OR ERR NEQ 0 THEN ERR:=RANKERROR ELSE
              IF RD.SPF NEQ 0 THEN IF BOOLEAN(RD.CHRMODE) THEN
                  BEGIN MOVEC(J,L+RD.SPF,K:=FINDSIZE(RD));J:=J+K;
                  END ELSE J:=MOVEN(J,L+RD.SPF,FINDSIZE(RD));
          IF ERR=0 THEN
              IF J=0 THEN SEMICOL:=NULLV ELSE
              BEGIN L:=GETSPACE((K:=ENTIER((J+7)DIV 8))+1);
              TRANSFERSP(INTO,SP,L+1,BUFFER,0,K);
              SP[LOC]:=J; SEMICOL:=L&1[CRF]&DDPUVC[CDID];
              END;

      END;
  BOOLEAN PROCEDURE SETUPLINE;
      BEGIN REAL T;INTEGER M;
      IF T:=ANALYZE(FALSE) NEQ 0 THEN % WE HAVE A PROGRAM DESC
          BEGIN IMS(3);
          M:=GETSPACE(1); SP[MOC]:=T;
          LASTMKS:=ST-STACKBASE;
          PUSH; IF ERR=0 THEN
              BEGIN AREG:=PROGMKS&LASTMKS[BACKPT]&1[CI]&M[SPTSP];
              POLLOC:=M:=T.SPF; POLTOP:=SP[MOC];
              LASTMKS:=LASTMKS+1+STACKBASE; CINDEX:=1;
              END;
          SETUPLINE:=TRUE;
          END ELSE SETUPLINE:=FALSE;
      END;
BOOLEAN PROCEDURE POPPROGRAM(OLDDATA,LASTMKS);
    REAL OLDDATA,LASTMKS;
    BEGIN LABEL EXIT;REAL L,M,N;
    WHILE TRUE DO
        BEGIN
        WHILE(L:=AREG).DATADESC NEQ 0 AND ERR=0 DO POP;
        IF L.DTD=PROGMKS THEN
            IF L=0 THEN %SOMETHING IS FUNNY...CONTINUE POPPING
            POP
            ELSE BEGIN
            LASTMKS:=M:=L.BACKF+STACKBASE;
            IF L.BACKF NEQ 0 AND NOT ((N:=SP[MOC]).DID=IMKS
               AND N.QUADIN=4) THEN POPPROGRAM:=TRUE;
            IF N.DID NEQ FMKS THEN
            FORGETPROGRAM(L);POP;GO TO EXIT;
            END ELSE %NOT A PROGRAM MKS
            IF L.DID=FMKS THEN
                BEGIN % MUST CUT BACK STATE VECTOR
                M:=STACKBASE+1;M:=SP[MOC].SPF+1;N:=SP[MOC].SPF+M;
                IF BOOLEAN(SP[NOC].SUSPENDED) THEN BEGIN SP[MOC].RF:=L:=
                    SP[MOC].RF-1;IF L=0 THEN SUSPENSION:=0;END;
                SP[NOC]:=0;SP[MOC].SPF:=N-M-1;POP;
                END ELSE % NOT A FMKS EITHER
                IF L.DID=IMKS THEN
                    BEGIN SCRATCHAIN(OLDDATA);OLDDATA:=L.SPF;POP;END;
        IF ERR NEQ 0 THEN GO TO EXIT;
        END; % OF THE DO
EXIT: END;%OF PROCEDURE POPPROGRAM
REAL PROCEDURE BUILDALPHA(LASTCONSTANT);
INTEGER LASTCONSTANT;
    BEGIN
    ARRAY B[0:BUFFSIZE];
    REAL R;
    INTEGER L,N;
    REAL STREAM PROCEDURE GETCHRS(ADDR,B); VALUE ADDR;
        BEGIN LOCAL C1,C2,TDI,TSI,QM;
        LOCAL ARROW;
        LABEL L,DSONE,FINIS,ERR;
        DI:=LOC QM; DS:=2RESET; DS:=2SET;
        DI:=LOC ARROW; DS:=RESET; DS:=7SET;
        DI:=B; DS:=8LIT"0";
        SI:=ADDR;
    L:
        IF SC=""" THEN % MAY BE A DOUBLE QUOTE
            BEGIN
            SI:=SI+1;
            IF SC=""" THEN % GET RID OF A QUOTE
                GO TO DSONE;
            COMMENT ELSE WE ARE LOOKING PAST THE RH QUOTE;
            GO TO FINIS;
            END ELSE % LOOK FOR THE QUESTION MARK
            BEGIN TDI:=DI; DI:=LOC QM;
            IF SC=DC THEN % END OF BUFFER ENCOUNTERED
                GO TO ERR;
            SI:=SI-1; DI:=LOC ARROW;
            IF SC=DC THEN %FOUND LEFT ARROW
                GO TO ERR;
            SI:=SI-1; DI:=TDI; GO TO DSONE
            END;
    DSONE: DS:=CHR; TALLY:=TALLY+1;
        C2:=TALLY; TSI:=SI; SI:=LOC C2; SI:=SI+7;
        IF SC="0" THEN
            BEGIN TALLY:=C1; TALLY:=TALLY+1; C1:=TALLY;
            TALLY:=0;
            END;
       SI:=TSI;
        GO TO L;
    FINIS: GETCHRS:=SI;
        DI:=B; SI:=LOC C1; SI:=SI+1; DS:=7CHR; SI:=LOC C2;
        SI:=SI+7; DS:=CHR;
    ERR:
    END;
    IF R:=GETCHRS(ADDRESS,B) NEQ 0 THEN % GOT A VECTOR
        IF NOT CURRENTMODE=FUNCMODE THEN
        BEGIN ADDRESS:=R;
        COMMENT B[0] HAS THE LENGTH OF THE STRING;
        IF R:=B[0] GEQ 1 THEN COMMENT A VECTOR;
            BEGIN
    L:=GETSPACE(N:=(R-1)DIV 8+2);
    TRANSFERSP(INTO,SP,L,B,0,N);
            SP[LOC]:=R;
            END;
      N:=GETSPACE(1);
      R:= L;
       R.DID:=DDPNVC;
    R.BACKP:=LASTCONSTANT;
   LASTCONSTANT:=N;
       IF B[0]=0 THEN R.DID:=DDPNVW %NULL BECAUSE .SPF=.RF=0
               %DON"T WANT CHARACTER NULL TO LOOK LIKE CHARS
           ELSE R.RF:=1;
      SP[NOC]:=R;
      COMMENT WE HAVE BUILT THE VECTOR AND DESCRIPTOR;
      BUILDALPHA:=N
      END
           ELSE BEGIN BUILDALPHA:=1;ADDRESS:=R END;
```

```
            %ELSE WE HAVE AN ERROR (MISSING " ETC.)
        END; % OF THE BUILD ALPHA PROCEDURE
  PROCEDURE PACK(L,OFFSET,N); VALUE L,OFFSET,N;
        INTEGER L,OFFSET,N;
        BEGIN
        LABEL QUIT;
        INTEGER M,T,MB,S;
        STREAM PROCEDURE PACKEM(A,B,N); VALUE N;
            BEGIN LOCAL T;
            SI:=LOC N; DI:=LOC T; DI:=DI+1; DS:=7CHR;
            SI:=A; DI:=B;
            T(2(32(SI:=SI+7; DS:=CHR))); N(SI:=SI+7; DS:=CHR);
            END;
         IF N = 0 THEN GO TO QUIT;
        T:=(M:=L:=L+OFFSET)+N;
        MB:=MAXBUFFSIZE DIV 8 × 8;
        WHILE M LSS T DO
            BEGIN
            TRANSFERSP(OUTOF,SP,M,BUFFER,0,MB:=MIN(MB,T-M));
            PACKEM(BUFFER,ACCUM,MB);
            TRANSFERSP(INTO,SP,L,ACCUM,0,S:=(MB+7)DIV 8);
            L:=L+S; M:=M+MB
            END;
        FORGETSPACE(L,T-L);
     QUIT:  END PROCEDURE PACK;
  INTEGER PROCEDURE UNPACK(S,OFFSET,N); VALUE N,S,OFFSET;
        INTEGER N,S,OFFSET;
        BEGIN
        INTEGER L,M,K,MB,T;
        LABEL QUIT;
        STREAM PROCEDURE UNPACKEM(A,B,N); VALUE N;
            BEGIN
            LOCAL T;
            SI:=LOC N; DI:=LOC T; DI:=DI+1; DS:=7CHR;
            SI:=A; DI:=B;
            T(2(32(DS:=7LIT"0"; DS:=CHR)));
            N(DS:=7LIT"0"; DS:=CHR);
            END;
        IF N = 0 THEN BEGIN UNPACK := S; GO TO QUIT; END;
        UNPACK:=L:=GETSPACE(OFFSET+N); K:=S+OFFSET-1;
        FOR M:=S STEP 1 UNTIL K DO
            BEGIN SP[LOC]:=SP[MOC]; L:=L+1
            END;
        K:=L+N; S:=S+OFFSET;
        MB:=MAXBUFFSIZE DIV 8;
        N := MB × 8;
        WHILE L LSS K DO
            BEGIN
            TRANSFERSP(OUTOF,SP,S,BUFFER,0,M:=MIN(MB,(K-L+7)DIV 8));
            UNPACKEM(BUFFER,ACCUM, M := MIN(K-L, M×8));
            TRANSFERSP(INTO,SP,L,ACCUM,0,M);
            L := L+N; S := S+MB
            END;
     QUIT:  END PROCEDURE UNPACK;
  PROCEDURE TRANSPOSE;
     BEGIN INTEGER M,N,L,I,ROW,COL,RANK,OUTER,INNER; REAL NEWDESC;
        INTEGER SIZE,J,MAT,TOP,START; BOOLEAN CHARACTER;
     LABEL QUIT; DEFINE GIVEUP=GO TO QUIT#;
     REAL NULL, DESC;
     DEFINE RESULT=RESULTD#;
     NULL := AREG; DESC := BREG;
     IF L:=DESC.SPF=0 THEN BEGIN ERR:=DOMAINERROR; GIVEUP; END;
     RANK := DESC.RF;
     SIZE := FINDSIZE(DESC);
     IF RANK LSS 2 THEN BEGIN NEWDESC:=DESC;
        %THEN THE TRANSPOSE IS THE THING ITSELF
              NEWDESC.NAMED:=0;
           NEWDESC.SPF := N:=GETSPACE(RANK+SIZE);
           SPCOPY(L,N,RANK+SIZE);
           GO TO QUIT; END;
        IF DESC.ARRAYTYPE=1 THEN BEGIN
          L:=UNPACK(L,RANK,SIZE);
          CHARACTER := TRUE; END;
        N:=L+RANK-1; COL := SP[NOC];
        N:=N-1; ROW := SP[NOC];
        TOP := SIZE DIV (MAT:=ROW×COL);
        NEWDESC := DESC;
        NEWDESC.SPF := M := GETSPACE(SIZE+RANK);
        SPCOPY (L,M,RANK-2);
        N:=M+RANK-1; SP[NOC]:=ROW;
        N:=N-1; SP[NOC] := COL;
        J:=0; M:=M+RANK;
      WHILE J LSS TOP DO BEGIN
      OUTER:=(START:=L+RANK+J×MAT) + COL - 1;
    FOR I:=START STEP 1 UNTIL OUTER DO BEGIN INNER:=I+MAT-1;
      FOR N:=I STEP COL UNTIL INNER DO
       BEGIN SP[MOC] := SP[NOC]; M:=M+1; END; END;
      J:=J+1; END;
   QUIT:  IF CHARACTER THEN BEGIN NEWDESC.ARRAYTYPE:=1;
         FORGETSPACE(L,SIZE+RANK);
         PACK(NEWDESC.SPF, RANK,SIZE); END;
    RESULTD := NEWDESC;
  END PROCEDURE TRANSPOSE;
 BOOLEAN PROCEDURE MATCHDIM(DESC1,DESC2); REAL DESC1,DESC2;
  BEGIN INTEGER I,L,M,TOP; LABEL DONE;
     MATCHDIM:= TRUE;
     IF DESC1.RF NEQ DESC2.RF THEN BEGIN MATCHDIM:=FALSE;
        ERR:=RANKERROR; GO TO DONE; END;
     I:=DESC1.SPF; M:=DESC2.SPF; TOP:=I+DESC1.RF-1;
     FOR L:=I STEP 1 UNTIL TOP DO BEGIN
       IF SP[LOC] NEQ SP[MOC] THEN BEGIN MATCHDIM:=FALSE;
        ERR:=LENGTHERROR; GO TO DONE; END;
     M:=M+1; END;
   DONE:  END PROCEDURE MATCHDIM;
 INTEGER PROCEDURE RANDINT(A,B,U); VALUE A,B;
   REAL A,B,U;
   BEGIN DEFINE QQMODUL = 67108864#, QQMULT = 8189#,
     QQRANDOM=(U:=U×QQMULT MOD QQMODUL)/QQMODUL#;
    RANDINT := (B-A+1)×QQRANDOM+A-.5;
  END PROCEDURE RANDINT;
 BOOLEAN PROCEDURE BOOLTYPE(A,B); REAL A,B;
   BEGIN IF ABS(A-1) LEQ FUZZ THEN A:=1;
     IF ABS(A) LEQ FUZZ THEN A:=0;
     IF ABS(B-1) LEQ FUZZ THEN B:=1;
     IF ABS(B) LEQ FUZZ THEN B:=0;
     BOOLTYPE := (IF A=1 OR A=0 AND B=1 OR B=0 THEN TRUE
        ELSE FALSE); END PROCEDURE BOOLTYPE;
 REAL PROCEDURE GAMMA(X); REAL X;
    COMMENT THIS PROCEDURE WAS TAKEN FROM ACM ALGORITHM 31.
      THE ONLY DIFFERENCE IS THAT THERE IS NO PROVISION FOR
      X LEQ 0 SINCE IT WILL NOT BE CALLED IN THAT CASE.  IT
      IS SUPPOSED TO GIVE ACCURACY TO 7 DIGITS;
   BEGIN REAL H,Y; LABEL A1, A2;
   H := 1; Y := X;
   A1:  IF Y = 2 THEN GO TO A2 ELSE IF Y LSS 2 THEN BEGIN
       H:=H/Y; Y:=Y+1; GO TO A1 END
     ELSE IF Y GEQ 3 THEN BEGIN
       Y:=Y-1; H:=H×Y; GO TO A1 END
     ELSE BEGIN Y := Y - 2;
       H := (((((((.0016063118 × Y + .0051589951) × Y
            + .0044511400) × Y + .0721101567) × Y
            + .0821117404) × Y + .4117741955) × Y
            + .4227874605) × Y + .9999999758) × H END;
   A2:  GAMMA := H;
  END OF PROCEDURE GAMMA;
 BOOLEAN PROCEDURE EXCLAM(MARG,NARG,M,ANS); VALUE MARG,NARG,M;
    REAL MARG,NARG,ANS; INTEGER M;
   BEGIN INTEGER N,I; REAL DENOM; LABEL PUT;
    EXCLAM := TRUE;
  IF I:=NARG.[1:8] NEQ 0 OR DENOM:=MARG.[1:8] NEQ 0 THEN BEGIN
    IF MARG LSS 0 OR NARG LSS 0 THEN BEGIN EXCLAM:=FALSE;
         GO TO PUT; END;
    IF M=0 THEN ANS:=GAMMA(NARG) ELSE BEGIN
      IF (NARG-MARG) LEQ 0 THEN BEGIN EXCLAM:=FALSE; GO TO PUT END;
      ANS := 1;
      IF I=0 THEN FOR I:=2 STEP 1 UNTIL NARG DO ANS:=ANS×I
          ELSE ANS:=GAMMA(NARG);
      IF DENOM=0 THEN BEGIN DENOM:=1; FOR I:=2 STEP 1 UNTIL MARG DO
         DENOM:=DENOM×I END ELSE DENOM:=GAMMA(MARG);
      ANS := ANS / (DENOM × GAMMA(NARG-MARG));
   END;
   GO TO PUT; END;
      IF M=0 THEN BEGIN ANS := 1;
      FOR I:=1 STEP 1 UNTIL NARG DO ANS:=ANS×I;
        GO TO PUT; END
       ELSE BEGIN IF MARG GTR NARG THEN
       BEGIN ANS:=0; GO TO PUT; END;
      IF MARG=0 THEN BEGIN ANS:=1; GO TO PUT; END;
      ANS := NARG - MARG + 1;
      FOR I:=NARG-MARG+2 STEP 1 UNTIL NARG DO ANS:=ANS×I;
      DENOM := 1;
      FOR I:=2 STEP 1 UNTIL MARG DO DENOM:=DENOM×I;
      ANS := ANS / DENOM; END;
  PUT:  END PROCEDURE EXCLAM;
 BOOLEAN PROCEDURE OPERATION(LEFT,RIGHT,LPTR,OP,ANS);
  COMMENT:  OP DEFINES THE APL OPERATORS AS FOLLOWS:
```

```
 OP   APL OPERATOR    OP   APL OPERATOR
 0         +          10      FACT-COMB
 1       TIMES        11          LSS
 2         -          12           =
 3        DIV         13          GEQ
 4         *          14          GTR
 5       RNDM         15          NEQ
 6     RESD-ABS       16          LEQ
 7     MIN-FLR        17          AND
 8     MAX-CEIL       18           OR
 9       NOT         19           NAND
                     20           NOR
                     21         LN-LOG
     THE "CIRCLE" OPERATORS FOLLOW.
 22       PI ×        30      SQRT(1-B*2)
 23     ARCTANH       31          SIN
 24     ARCCOSH       32          COS
 25     ARCSINH       33          TAN
 26     SQRT(B*2-1)   34         SQRT(1+B*2)
 27     ARCTAN        35          SINH
 28     ARCCOS        36          COSH
 29     ARCSIN        37          TANH;

 COMMENT:  LPTR IS LSS 0 IF THE CALL COMES FROM A
                             REDUCTION TYPE PROCEDURE.
          LPTR = 0 IF OPERATOR IS MONADIC.
            LPTR GTR 0 IF OPERATOR IS DYADIC.
            LPTR LSS 0 IF COMES FROM REDUCTION TYPE OPERATION;
      VALUE LEFT,RIGHT,LPTR,OP;
    REAL LEFT,RIGHT,LPTR,OP;
  REAL ANS;
BEGIN LABEL PUT,DOMAIN,KITE; DEFINE GIVEUP=GO TO PUT#;
 DEFINE MAXEXP=158.037557167#,
         MINEXP=-103.7216898#;
 MONITOR INTOVR, ZERO, EXPOVR;
 OPERATION := TRUE;
  IF LPTR LSS 0 THEN IF OP GTR 10 AND OP LSS 21 THEN
        IF NOT BOOLTYPE(LEFT,RIGHT) THEN GO TO DOMAIN;
  IF OP = 45 THEN IF LPTR=0 THEN OP:=22
   ELSE IF ABS(LEFT) GTR 7 THEN GO TO DOMAIN
     ELSE OP := LEFT + 30;
  IF OP GTR 16 AND OP LSS 21 THEN IF NOT BOOLTYPE(LEFT,RIGHT)
       THEN GO TO DOMAIN;
  ZERO:=DOMAIN; INTOVR:=KITE; EXPOVR:=KITE;
    CASE OP OP BEGIN
  ANS := LEFT + RIGHT;
  ANS := IF LPTR=0 THEN SIGN(RIGHT) ELSE LEFT × RIGHT;
  ANS := LEFT - RIGHT;
  ANS := LEFT / RIGHT;
 IF LPTR=0 THEN IF RIGHT GTR MINEXP AND RIGHT LSS MAXEXP
                      THEN ANS:=EXP(RIGHT) ELSE GO TO KITE
    ELSE IF RIGHT.[3:6]=0 THEN ANS:=LEFT*ENTIER(RIGHT)
       ELSE IF LEFT GTR 0 THEN IF ANS:=RIGHT×LN(LEFT) GTR MINEXP
                                  AND ANS LSS MAXEXP THEN
                              ANS:=EXP(ANS) ELSE GO TO KITE
            ELSE IF LEFT=0 AND RIGHT GTR 0 THEN ANS:=0
                 ELSE GO TO DOMAIN;
  IF LPTR NEQ 0 THEN BEGIN ERR:=SYSTEMERROR; GIVEUP; END ELSE
 IF RIGHT LSS ORIGIN THEN GO TO DOMAIN ELSE
    ANS := RANDINT(ORIGIN,RIGHT,SEED);
  IF LPTR=0 THEN ANS := ABS(RIGHT) ELSE
    BEGIN IF LEFT=0 THEN IF RIGHT GEQ 0 THEN
   ANS := RIGHT ELSE GO TO DOMAIN
     ELSE IF (ANS:=RIGHT MOD LEFT) LSS 0
       THEN ANS:=ANS + ABS(LEFT); END;
  ANS := (IF LPTR=0 THEN ENTIER(RIGHT+FUZZ)
     ELSE IF LEFT LEQ RIGHT THEN LEFT ELSE RIGHT);
  ANS := (IF LPTR=0 THEN -ENTIER(-RIGHT+FUZZ)
     ELSE IF LEFT GTR RIGHT THEN LEFT ELSE RIGHT);
  IF LPTR NEQ 0 THEN BEGIN ERR:=SYNTAXERROR; GIVEUP; END
     ELSE IF NOT BOOLTYPE(0,RIGHT) THEN
      BEGIN ERR:=DOMAINERROR; GIVEUP; END
    ELSE ANS := (IF RIGHT=1 THEN 0 ELSE 1);
 IF NOT EXCLAM(LEFT,RIGHT,LPTR,ANS) THEN GO TO DOMAIN;

  ANS := (IF RIGHT-LEFT GTR FUZZ×ABS(RIGHT) THEN 1 ELSE 0);
  ANS:=(IF ABS(LEFT-RIGHT) LEQ FUZZ×ABS(RIGHT) THEN 1 ELSE 0);
  ANS:=(IF RIGHT-LEFT LEQ FUZZ×ABS(RIGHT) THEN 1 ELSE 0);
  ANS:=(IF LEFT-RIGHT GTR FUZZ×ABS(RIGHT) THEN 1 ELSE 0);
 ANS:=(IF ABS(LEFT-RIGHT) GTR FUZZ×ABS(RIGHT) THEN 1 ELSE 0);
  ANS:=(IF LEFT-RIGHT LEQ FUZZ×ABS(RIGHT) THEN 1 ELSE 0);
  ANS := RIGHT × LEFT;  %AND
  ANS := IF RIGHT + LEFT = 0 THEN 0 ELSE 1;  %OR
```

```
    ANS := IF  RIGHT × LEFT = 1 THEN 0 ELSE 1;  %NAND
     ANS := IF RIGHT + LEFT = 0 THEN 1 ELSE 0; %NOR
   IF RIGHT LEQ 0 THEN GO TO DOMAIN ELSE IF LPTR=0 THEN
        ANS:=LN(RIGHT) ELSE
      IF LEFT LEQ 1 THEN GO TO DOMAIN ELSE
        ANS := LN(RIGHT) / LN(LEFT);              %LOGARITHMS
    ANS := 3.1415926536 × RIGHT;
    IF ABS(RIGHT) GEQ 1 THEN GO TO DOMAIN ELSE
      ANS:= .5×LN((1+RIGHT)/(1-RIGHT));  %ARCTANH

    IF RIGHT LSS 1 THEN GO TO DOMAIN ELSE
     ANS:=LN(RIGHT+SQRT(RIGHT×RIGHT-1));  %ARCCOSH
   ANS := LN(RIGHT + SQRT(RIGHT×RIGHT+1));  %ARCSINH

    IF ABS(RIGHT) LSS 1 THEN GO TO DOMAIN ELSE
     ANS:=SQRT(RIGHT×RIGHT-1);
    ANS := ARCTAN(RIGHT);
   IF ABS(RIGHT) GTR 1 THEN GO TO DOMAIN ELSE
       IF RIGHT=0 THEN ANS:=1.5707963268 ELSE
     ANS:=ARCTAN(SQRT(1-RIGHT*2)/RIGHT);        %ARCCOS
    IF ABS(RIGHT) GEQ 1 THEN GO TO DOMAIN ELSE
       ANS:=ARCTAN(RIGHT/ SQRT(1-RIGHT*2));     %ARCSIN
    IF ABS(RIGHT) GTR 1 THEN GO TO DOMAIN ELSE
       ANS := SQRT(1-RIGHT*2);
    ANS := SIN(RIGHT);
    ANS := COS(RIGHT);
    ANS := SIN(RIGHT) / COS(RIGHT);        %TAN
   ANS := SQRT(1+RIGHT×RIGHT);
    ANS := (EXP(RIGHT) - EXP(-RIGHT))/2;    %SINH
    ANS := (EXP(RIGHT) + EXP(-RIGHT))/2;    %COSH
    ANS := ((OP:=EXP(RIGHT))-(ANS:=EXP(-RIGHT)))/(OP+ANS); %TANH
   END;
 GO TO PUT;
 KITE:  ERR:=KITEERROR; GO TO PUT;
  DOMAIN:  ERR:=DOMAINERROR;
  PUT:  IF ERR NEQ 0 THEN OPERATION := FALSE;
    END PROCEDURE OPERATION;
PROCEDURE ARITH(OP); VALUE OP;
      INTEGER OP;
  COMMENT:  ARITH HANDLES ALL APL OPERATORS THAT EMPLOY THE
     VECTOR-VECTOR, SCALAR-VECTOR, SCALAR-SCALAR, VECTOR-SCALAR
     FEATURE.  DESC1 AND DESC2 ARE THE DESCRIPTORS FOR THE
     LEFTHAND AND RIGHTHAND OPERANDS, RESPECTIVELY.  IF
     IF DESC1 = 0, THE OPERATOR IS TAKEN TO BE MONADIC.
     IF DESC.SPF = 0, THE OPERAND IS NULL AND A DOMAIN ERROR
     RESULTS EXCEPT IN THE CASE OF MULTIPLICATION.
     OP IS AN INTERNAL OPERATION CODE FOR THE OPERATOR, WHICH
     DEPENDS ON THE CASE STATEMENT IN THE OPERATION PROCEDURE.;
BEGIN INTEGER L,M,I,N,SIZE,RANK1,RANK2,TOP,
        FORGETL, FORGETM;
    REAL DESC,LEFT,RIGHT,ANS,SIZE1,SIZE2,DESC1,DESC2;
    LABEL DONE, LEFTSCALE, SCALVECT, DOMAIN, VECTSCAL;
  BOOLEAN CHAR1, CHAR2;
  DESC1 := AREG; DESC2 := BREG;
  L:=DESC1.SPF; M:=DESC2.SPF;
  RANK1:=DESC1.RF; RANK2:=DESC2.RF;
  SIZE1:=FINDSIZE(DESC1);  SIZE2:=FINDSIZE(DESC2);
  IF(CHAR1:=DESC1.ARRAYTYPE=1) OR (CHAR2:=DESC2.ARRAYTYPE=1)
  THEN BEGIN IF OP LSS 11 OR OP GTR 16
     OR NOT(CHAR1 AND CHAR2) AND NOT(OP=12 OR OP=15)
      THEN BEGIN CHAR1:=CHAR2:=FALSE; GO TO DOMAIN; END;
    IF CHAR1 THEN
        FORGETL := L := UNPACK(L,RANK1,SIZE1);
    IF CHAR2 THEN
        FORGETM := M := UNPACK(M,RANK2,SIZE2); END;


    IF M=0 THEN BEGIN IF OP NEQ 1 THEN GO TO DOMAIN
                   ELSE BEGIN DESC := NULLV;
                                    GO TO DONE; END;  END;
  IF L=0 THEN BEGIN
  IF DESC1.DID NEQ 0 THEN
    IF OP=1 THEN BEGIN DESC:=NULLV; GO TO DONE; END
      ELSE GO TO DOMAIN;
    IF OP GTR 10 AND OP LSS 21 THEN GO TO DOMAIN;
     LEFT := OP MOD 2; GO TO LEFTSCALE; END;
   IF SIZE1=1
       THEN BEGIN L:=L+RANK1; LEFT:=SP[LOC];
     GO TO LEFTSCALE; END;
   IF SIZE2=1 THEN BEGIN
     % DESC1 IS A VECTOR, DESC2 IS A SCALAR;
  VECTSCAL:  M:=M+RANK2; RIGHT:=SP[MOC];
   I := GETSPACE( SIZE:=SIZE1+RANK1);
```

```
    DESC.SPF:=I; DESC.DID:=DDPUVW; SPCOPY(L,I,RANK1);
    L:=L+RANK1; I:=I+RANK1;
    DESC.RF:=RANK1;  TOP:=SIZE1+I-1;
    FOR N:=I STEP 1 UNTIL TOP DO BEGIN
      IF OPERATION(SP[LOC],RIGHT,L,OP,ANS) THEN
          SP[NOC] := ANS ELSE GO TO DONE;
      L:=L+1; END;
    GO TO DONE; END;
 % BOTH DESC1 AND DESC2 ARE ARRAYS;
   IF NOT MATCHDIM(DESC1,DESC2) THEN GO TO DONE
      ELSE BEGIN
      I := GETSPACE( SIZE := SIZE2 + RANK2 );
      SPCOPY(M,I,RANK2); DESC.SPF:=I; DESC.DID:=DDPUVW;
      DESC.RF := RANK2;
      M:=M+RANK2; I:=I+RANK2; L:=L+RANK2;
      TOP := I+SIZE2-1;
      FOR N:=I STEP 1 UNTIL TOP DO BEGIN
      IF OPERATION(SP[LOC],SP[MOC],L,OP,ANS) THEN
         SP[NOC] := ANS ELSE GO TO DONE;
        L:=L+1; M:=M+1; END;
      GO TO DONE; END;
    LEFTSCALE:  IF SIZE2 = 1
           THEN BEGIN
  IF RANK1 NEQ RANK2 THEN BEGIN
      IF RANK1=0 THEN GO TO SCALVECT;
       IF RANK2=0 THEN BEGIN L:=L-RANK1; GO TO VECTSCAL; END;
    IF CHAR1 AND RANK1=1 THEN GO TO SCALVECT;
    IF CHAR2 AND RANK2=1 THEN GO TO VECTSCAL;
      ERR:=KITEERROR; GO TO DONE; END
     ELSE IF RANK1xRANK2 NEQ 0 THEN GO TO SCALVECT;
  % BOTH OPERANDS ARE SCALAR;
    M := M + RANK2;
    N := GETSPACE(SIZE:=1); RIGHT:=SP[MOC];
     DESC.SPF := N; DESC.DID := DDPUSW;
     IF OPERATION(LEFT,RIGHT,L,OP,ANS) THEN
       SP[NOC] := ANS ELSE GO TO DONE;
     GO TO DONE; END
    ELSE BEGIN %DESC1 IS SCALAR, DESC2 IS VECTOR;

  SCALVECT:  I := GETSPACE( SIZE := SIZE2 + RANK2);
     DESC.SPF := I; DESC.RF := RANK2; DESC.DID:=DDPUVW;
     SPCOPY(M,I,RANK2);
      M:=M+RANK2; I:=I+RANK2; TOP:=SIZE2+I-1;
     FOR N:=I STEP 1 UNTIL TOP DO BEGIN
       IF OPERATION(LEFT,SP[MOC],L,OP,ANS)
        THEN SP[NOC] := ANS ELSE GO TO DONE;
       M := M+1; END;
      END;
    GO TO DONE;
   DOMAIN:  ERR := DOMAINERROR;
  DONE:  RESULTD := DESC;
    IF CHAR1 THEN FORGETSPACE(FORGETL,SIZE1+RANK1);
    IF CHAR2 THEN FORGETSPACE(FORGETM,SIZE2+RANK2);
  IF ERR NEQ 0 THEN FORGETSPACE(DESC.SPF, SIZE);
   END PROCEDURE ARITH;
 PROCEDURE DYADICRNDM;
   BEGIN INTEGER NUM, KIND; REAL DESC;
   REAL DESC1, DESC2;
   INTEGER L,M,N,T,I,TEMP,OUTTOP,TOP,PICK; LABEL QUIT;
   INTEGER START; LABEL INSERT;
     DESC1 := AREG; DESC2 := BREG;
     IF FINDSIZE(DESC1) NEQ 1 OR FINDSIZE(DESC2) NEQ 1
       THEN BEGIN ERR:=RANKERROR; GO TO QUIT; END;
     IF DESC1.SPF=0 OR DESC2.SPF=0 THEN BEGIN
              ERR:=DOMAINERROR; GO TO QUIT; END;
     L:=DESC1.SPF+DESC1.RF; M:=DESC2.SPF+DESC2.RF;
     NUM := SP[LOC]; KIND := SP[MOC];
     IF KIND LSS ORIGIN
      OR NUM GTR PICK := KIND-ORIGIN+1
      OR DESC1.ARRAYTYPE=1
      OR DESC2.ARRAYTYPE=1 THEN BEGIN ERR:=DOMAINERROR;
       GO TO QUIT; END;
     DESC.DID := DDPUVW; DESC.RF := 1;
   IF NUM LEQ 0 THEN BEGIN DESC := NULLV; GO TO QUIT; END;
   IF NUM GTR MAXWORDSTORE THEN BEGIN ERR:=KITEERROR; GO TO QUIT END;
     DESC.SPF := L := GETSPACE(NUM+1);
     SP[LOC] := NUM; L := L+1;
     OUTTOP := L+NUM-1;
   TEMP := GETSPACE(NUM);
   START:=ORIGIN; I:=0;
   FOR L:=L STEP 1 UNTIL OUTTOP DO BEGIN
     PICK:=RANDINT(START,KIND,SEED);
     M:=TEMP;
```

```
       IF I = 0 OR PICK LSS SP[MOC] THEN N:=TEMP
          ELSE BEGIN TOP:=TEMP+I-1;
             N:=TEMP+T:=I DIV 2;
             WHILE T GTR 0 DO
                IF PICK GEQ SP[NOC] THEN N:=N+T:=T DIV 2
                               ELSE N:=N-T:=T DIV 2;

             FOR N:=MAX(TEMP,N-3) STEP 1 UNTIL TOP DO
                IF SP[NOC] GTR PICK THEN
                      GO TO INSERT;
          END;
       INSERT:  IF L LSS OUTTOP THEN BEGIN TOP:=N+1; N:=TEMP+I;
        FOR M:=N STEP -1 UNTIL TOP DO BEGIN
           N:=N-1;  SP[MOC] := SP[NOC] - 1; END;
        SP[NOC] := PICK; END;
       SP[LOC] := N - TEMP + PICK;
       KIND:=KIND-1;
       I:=I+1;
       END;
    FORGETSPACE(TEMP,NUM);
    QUIT:  RESULTD := DESC;
    END PROCEDURE DYADICRNDM;
 PROCEDURE RHOP;
    BEGIN INTEGER RANK,M,POINT; REAL NEWDESC,DESC1,DESC;
    LABEL QUIT, WORK; BOOLEAN CHARACTER;
    DEFINE TOOBIG=BEGIN ERR:=KITEERROR; GO TO QUIT; END#;
    INTEGER N,TOP,NEWRANK,RANK1, POINT1,SIZE1,L,SIZE2;
    DESC1 := AREG; DESC := BREG;
    IF DESC.SPF = 0 THEN BEGIN ERR:=DOMAINERROR; GO TO QUIT; END;
    IF DESC1.DID NEQ 0 THEN BEGIN %--DYADIC RHO--RESTRUCTURING---------
       IF L:=DESC1.SPF = 0 THEN BEGIN %NULL LEFT OP MEANS SCALAR ANS
          IF DESC.ARRAYTYPE=1 THEN TOOBIG; %NO SCALAR CHARACTERS
          NEWDESC.SPF:=M:=GETSPACE(1);
          NEWDESC.DID:=DDPUSW;
          L:=DESC.SPF+DESC.RF;
          SP[MOC]:=SP[LOC]; GO TO QUIT; END;
       IF DESC1.ARRAYTYPE NEQ 0 THEN BEGIN
             ERR:=DOMAINERROR; GO TO QUIT; END;
       RANK1:=DESC1.RF;
       IF FINDSIZE(DESC1)=1 THEN BEGIN
          N:=L+RANK1;
          IF SIZE1:=ENTIER(SP[NOC]+.5) LSS 0 THEN BEGIN
                ERR:=DOMAINERROR; GO TO QUIT; END;
          NEWRANK:=1; TOP:=N; GO TO WORK; END;
       IF RANK1 NEQ 1 THEN BEGIN ERR:=RANKERROR; GO TO QUIT; END;
       IF NEWRANK:=SP[LOC] GTR 31 THEN TOOBIG;
       SIZE1:=1; TOP := L+NEWRANK+RANK1-1;
       IF NEWRANK LEQ 0 THEN BEGIN ERR:=SYSTEMERROR; GO TO QUIT; END;
       FOR N:=L+RANK1 STEP 1 UNTIL TOP DO
            IF SIZE1:=SIZE1×ENTIER(SP[NOC]+.5) LSS 0 THEN BEGIN
                ERR:=DOMAINERROR; GO TO QUIT; END;
 WORK:  IF SIZE1=0 THEN BEGIN NEWDESC := NULLV; GO TO QUIT END;
    IF SIZE1 GTR MAXWORDSTORE THEN TOOBIG;
    NEWDESC.DID:=DDPUVW; NEWDESC.RF:=NEWRANK;
    NEWDESC.SPF := M := GETSPACE(SIZE1+NEWRANK);
  %CANT USE SPCOPY FOR DIM VECTOR AS LEFT OP MAY NOT BE INTEGER
   FOR L:=L+RANK1 STEP 1 UNTIL TOP DO
       BEGIN SP[MOC]:=ENTIER(SP[LOC]+.5); M:=M+1; END;
       SIZE2:=FINDSIZE(DESC); L:=DESC.SPF; RANK:=DESC.RF;
       IF DESC.ARRAYTYPE=1 THEN BEGIN L:=UNPACK(L,RANK,SIZE2);
          CHARACTER:=TRUE; END; TOP:=SIZE1 DIV SIZE2; POINT:=L+RANK;
          FOR N:=1 STEP 1 UNTIL TOP DO BEGIN SPCOPY(POINT,M,SIZE2);
                M := M+SIZE2; END;
          TOP := SIZE1 MOD SIZE2; SPCOPY(POINT,M,TOP);
          GO TO QUIT; END ELSE
 %---------MONADIC RHO-----DIMENSION VECTOR--------------------
       RANK := DESC.RF; POINT := DESC.SPF;
       NEWDESC.DID := DDPUVW; NEWDESC.RF := 1;
       IF DESC.DATATYPE = 1 THEN BEGIN
          NEWDESC := NULLV; GO TO QUIT END;
       NEWDESC.SPF := M := GETSPACE(RANK+1);
       SP[MOC] := RANK;
       SPCOPY(POINT,M+1, RANK);
   QUIT:  IF CHARACTER THEN BEGIN NEWDESC.ARRAYTYPE:=1;
              FORGETSPACE(L,SIZE2+RANK);
              PACK(NEWDESC.SPF, NEWRANK,SIZE1); END;
      RESULTD := NEWDESC;
   END PROCEDURE RHOP;
 PROCEDURE IOTAP;
   BEGIN INTEGER I,L,M,TOP; REAL DESC;
   REAL LEFTOP, RIGHTOP;
   INTEGER RSIZE,LSIZE,RRANK,LRANK,N,LL,MM,TIP,NIX;
   LABEL QUIT, DONE;
   LEFTOP:=AREG; RIGHTOP:=BREG;
   IF L:=RIGHTOP.SPF=0 THEN BEGIN ERR:=DOMAINERROR; GO TO QUIT END;
   RSIZE:=FINDSIZE(RIGHTOP); RRANK:=RIGHTOP.RF;
   DESC.DID := DDPUVW; DESC.RF := 1;
   IF LEFTOP.DID NEQ 0 THEN BEGIN  %-------DYADIC IOTA------------
    IF LRANK := LEFTOP.RF GTR 1 THEN BEGIN ERR:=RANKERROR;
                            GO TO QUIT; END;
    LSIZE := FINDSIZE(LEFTOP);
    IF M:=LEFTOP.SPF=0 THEN BEGIN %RESULT IS ORIGIN IF IT WAS NULL
       DESC.SPF:=M:=GETSPACE(1); DESC.RF:=0; DESC.SCALAR:=1;
       SP[MOC] := ORIGIN; GO TO QUIT; END;
   IF LEFTOP.ARRAYTYPE=1 THEN M:=UNPACK(M,LRANK,LSIZE);
   IF RIGHTOP.ARRAYTYPE=1 THEN L:=UNPACK(L,RRANK,RSIZE);
    TIP := (NIX:=LSIZE+ORIGIN) - 1;
    DESC.SPF:=N:=GETSPACE(RSIZE+RRANK);
   IF RRANK=0 THEN DESC.SCALAR:=1 ; DESC.RF:=RRANK;
    SPCOPY(L,N,RRANK);
    MM := M+LRANK; LL:=L:=L+RRANK;
     TOP:=N+RRANK+RSIZE-1;
    FOR N:=N+RRANK STEP 1 UNTIL TOP DO BEGIN
       SP[NOC] := NIX;
       M := MM;
       FOR I:=ORIGIN STEP 1 UNTIL TIP DO
        IF OPERATION(SP[MOC],SP[LOC],1,12,LEFTOP) AND LEFTOP=1
          THEN BEGIN SP[NOC]:=I; GO TO DONE;
            END ELSE M:=M+1;
       DONE:  L:=L+1;  END;
   IF LEFTOP.ARRAYTYPE=1 THEN FORGETSPACE(MM-LRANK,LRANK+LSIZE);
   IF RIGHTOP.ARRAYTYPE=1 THEN FORGETSPACE(LL-RRANK,RRANK+RSIZE);
   END ELSE BEGIN %-------------MONADIC IOTA------------------
   IF RIGHTOP.ARRAYTYPE=1 THEN
        BEGIN ERR:=DOMAINERROR; GO TO QUIT
          END;
   IF RSIZE NEQ 1 THEN BEGIN ERR:=RANKERROR; GO TO QUIT END;

   L := L + RRANK;
   IF TOP:=SP[LOC] GTR MAXWORDSTORE THEN
      BEGIN ERR:=KITEERROR; GO TO QUIT
      END;

   IF TOP LSS ORIGIN THEN BEGIN DESC:=NULLV; GO TO QUIT END;
   DESC.SPF := M := GETSPACE(TOP+1);
   SP[MOC] := TOP; M := M+1;
   TOP := TOP + ORIGIN - 1;
   FOR I := ORIGIN STEP 1 UNTIL TOP DO BEGIN
      SP[MOC] := I; M := M+1;  END;
   END;
 QUIT:  RESULTD := DESC;
  END PROCEDURE IOTAP;
 PROCEDURE COMMAP;
   BEGIN REAL LDESC, RDESC;
     INTEGER L,M,N,LRANK,RRANK,LSIZE,RSIZE,SIZE;
    REAL DESC; LABEL QUIT; BOOLEAN CHARACTER;
     LDESC := AREG; RDESC := BREG;
    RRANK := RDESC.RF;  LRANK := LDESC.RF;
    LSIZE := IF (L := LDESC.SPF) = 0 THEN 0 ELSE FINDSIZE(LDESC);
    RSIZE := IF (M := RDESC.SPF) = 0 THEN 0 ELSE FINDSIZE(RDESC);
    IF RDESC.ARRAYTYPE = 1 THEN BEGIN
         M := UNPACK(M,RRANK,RSIZE);
         CHARACTER :=  TRUE;  END;
    DESC.DID := DDPUVW; DESC.RF := 1;
    IF LDESC.DID = 0 THEN BEGIN  %-----MONADIC COMMA--RAVEL--------
     IF RSIZE=0 THEN BEGIN DESC:=NULLV; GO TO QUIT END;
     DESC.SPF := L := GETSPACE(RSIZE+1);
     SP[LOC] := RSIZE;
     SPCOPY(M+RRANK, L+1, RSIZE);
     N := L; SIZE := RSIZE;
     GO TO QUIT; END
    ELSE BEGIN
   %HERE IS THE CODE FOR DYADIC COMMA, I.E. CATENATION
     IF RRANK NEQ 1 AND RSIZE GTR 1 OR
        LRANK NEQ 1 AND LSIZE GTR 1 THEN BEGIN
          ERR:= RANKERROR; GO TO QUIT; END;
   IF SIZE:=LSIZE+RSIZE GTR MAXWORDSTORE THEN BEGIN
          ERR:=KITEERROR; GO TO QUIT; END;
    COMMENT CANT MIX NUMBERS AND CHARACTERS. HAVE TO JUGGLE
         IF LEFT IS NUMBERS AND RIGHT IS CHARACTERS AS RIGHT
         HAS ALREADY BEEN UNPACKED AND WE DONT WANT TO FORGET
         LEFT AND WE DONT WANT TO PACK THE NON-RESULT;
    IF CHARACTER THEN
       IF LDESC.ARRAYTYPE=1 OR LSIZE=0 THEN L:=UNPACK(L,LRANK,LSIZE)
            ELSE BEGIN SIZE:=0; LSIZE:=-LRANK; ERR:=DOMAINERROR;
```

```
              GO TO QUIT END
   ELSE IF LDESC.ARRAYTYPE=1 THEN
      IF RSIZE NEQ 0 THEN
         BEGIN ERR:=DOMAINERROR; GO TO QUIT END
      ELSE BEGIN CHARACTER:=TRUE;
                 L:=UNPACK(L,LRANK,LSIZE); END;
   IF SIZE=0 THEN BEGIN DESC:=NULLV; GO TO QUIT END;
   DESC.SPF := N := GETSPACE(SIZE+1);
   SP[NOC] := SIZE;
      SPCOPY(L+LRANK, N+1, LSIZE);
      SPCOPY(M+RRANK, N+LSIZE+1, RSIZE);
   END;
   QUIT:
   IF CHARACTER THEN BEGIN DESC.ARRAYTYPE := 1;
      PACK(N,1,SIZE);
      FORGETSPACE(L,LSIZE+LRANK);
      FORGETSPACE(M,RSIZE+RRANK);
        END;
   RESULTD := DESC;
  END PROCEDURE COMMAP;
INTEGER STREAM PROCEDURE GETOP(A,N); VALUE N;
  BEGIN SI := A; SI := SI + N;
    DI := LOC GETOP;
    DS := 7 LIT "0"; DS := CHR;
  END PROCEDURE GETOP;
  REAL PROCEDURE IDENTITY(OP); VALUE OP; INTEGER OP;
  BEGIN
  CASE OP OF BEGIN
      IDENTITY := 0;         %FOR +
      IDENTITY := 1;         %FOR ×
      IDENTITY := 0;         %FOR -
      IDENTITY := 1;         %FOR DIV
      IDENTITY := 1;         %FOR *
                   ;         %NO REDUCTION ON RNDM
      IDENTITY := 0;         %FOR RESD
    IDENTITY := BIGGEST;  %FOR MIN
    IDENTITY := -BIGGEST; %FOR MAX
                    ;         %NOT ISNT DYADIC
      IDENTITY := 1;          %FOR COMB
     IDENTITY := 0;          %FOR LSS
     IDENTITY := 1;          %FOR =
     IDENTITY := 1;         %FOR GEQ
     IDENTITY := 0;          %FOR GTR
     IDENTITY := 0;          %FOR NEQ
     IDENTITY := 1;          %FOR LEQ
      IDENTITY := 1;          %FOR AND
      IDENTITY := 0;          %FOR OR
   END;   END PROCEDURE IDENTITY;
INTEGER PROCEDURE GETT(ALONG,RANK); VALUE ALONG, RANK;
      INTEGER ALONG, RANK;
  GETT := IF ALONG=1 THEN 0 ELSE
          IF ALONG=RANK THEN 2 ELSE
          IF ALONG=RANK-1 THEN 1 ELSE 0;
BOOLEAN PROCEDURE CHECKANDADD(SIZE,L,SUM);
    VALUE SIZE,L; INTEGER SIZE,L,SUM;
  BEGIN LABEL QUIT; INTEGER I,TOP,M,S,T;
    CHECKANDADD:=TRUE;
    SUM := 0;
    TOP := SIZE DIV 2 × 2 - 1 + L;
    FOR L:=L STEP 2 UNTIL TOP DO BEGIN  M:=L+1;
      IF NOT BOOLTYPE(S:=SP[LOC], T:=SP[MOC]) THEN BEGIN
           CHECKANDADD:=FALSE; GO TO QUIT; END
         ELSE SUM := SUM+S+T; END;
    IF SIZE MOD 2 = 1 THEN BEGIN
      IF NOT BOOLTYPE(T:=SP[LOC],0) THEN
        CHECKANDADD := FALSE ELSE SUM := SUM+T;
      END;
  QUIT:  END PROCEDURE CHECKANDADD;
PROCEDURE COMPRESS(LDESC, RDESC, DIM); VALUE LDESC,RDESC,DIM;
     REAL LDESC, RDESC, DIM;
  BEGIN INTEGER I,J,K,L,M,N,T,RANK,LSIZE,RSIZE,ALONG,TOP,
          FACTOR,SUM,DIMMOD,SIZE,LEFT,RIGHT,S;
    REAL DESC; BOOLEAN CHARACTER;
    LABEL QUIT,RANKE,DOMAIN,IDENT;
  DESC.DID := DDPUVW;
  IF L := LDESC.SPF = 0 THEN GO TO DOMAIN;
  IF M:=RDESC.SPF=0 THEN BEGIN DESC:=NULLV; GO TO QUIT; END;
  LSIZE := FINDSIZE(LDESC); RSIZE := FINDSIZE(RDESC);
  IF RANK:=LDESC.RF NEQ 1 THEN IF LSIZE NEQ 1
     THEN GO TO DOMAIN;
  LEFT := L := L+RANK;
  RANK := RDESC.RF;
  IF N:=DIM.SPF=0 AND DIM.DID NEQ 0 OR DIM.ARRAYTYPE=1
```

```
    OR LDESC.ARRAYTYPE=1 THEN GO TO DOMAIN;
  IF J:=DIM.RF NEQ 0 THEN BEGIN
     IF FINDSIZE(DIM)=1 THEN N:=N+J ELSE GO TO DOMAIN END;
  IF ALONG:=(IF N=J THEN RANK ELSE SP[NOC]-ORIGIN+1) GTR RANK
    OR ALONG LSS 1 AND RANK NEQ 0
    THEN BEGIN ERR:=INDEXERROR; GO TO QUIT; END;
  IF RANK = 0 THEN
    IF LSIZE NEQ 1 THEN GO TO DOMAIN ELSE BEGIN
      IF TOP:=SP[LOC]=0 THEN BEGIN DESC:=NULLV; GO TO QUIT; END;
      IF TOP = 1 THEN BEGIN DESC.SPF := N := GETSPACE(2);
          DESC.RF := SP[NOC] := 1;
          N:=N+1; SP[NOC]:=SP[MOC]; GO TO QUIT;
        END ELSE GO TO DOMAIN; END;
   IF LSIZE = 1 THEN BEGIN
     COMMENT IF LEFT ARG IS SCALAR, ANSWER IS NULL IF 0,
             RIGHT ARG IF 1;
    SUM:=SP[LOC];
    IF SUM NEQ 0 AND SUM NEQ 1 THEN GO TO DOMAIN

        ELSE GO TO IDENT; END;
  N := M+ALONG - 1;
  IF LSIZE NEQ (T:=SP[NOC]) THEN BEGIN
    ERR:=LENGTHERROR; GO TO QUIT; END;
  IF NOT CHECKANDADD(LSIZE,LEFT,SUM) THEN GO TO DOMAIN;
    IDENT:  IF SUM=0 THEN BEGIN DESC:=NULLV; GO TO QUIT END;
    IF SUM = LSIZE THEN BEGIN
      IF RDESC.ARRAYTYPE=1 THEN BEGIN
        RSIZE:=RSIZE DIV 8 + (IF RSIZE MOD 8 NEQ 0 THEN 1 ELSE 0);
        DESC.CHRMODE:=1; END;
      DESC.SPF:=N:=GETSPACE(TOP:=RSIZE+RANK);
      DESC.RF := RANK; SPCOPY(M,N,TOP); GO TO QUIT; END;
  SIZE := RSIZE DIV T × SUM;
  DESC.RF:=RANK;
  IF RDESC.ARRAYTYPE = 1 THEN BEGIN M:=UNPACK(M,RANK,RSIZE);
            CHARACTER := TRUE; END;
  RIGHT := M;
  DESC.SPF := S := GETSPACE(SIZE+RANK);
  N := S;
  FOR I:=1 STEP 1 UNTIL RANK DO BEGIN
    IF I=ALONG THEN SP[NOC]:=SUM ELSE SP[NOC]:=SP[MOC];
    N:=N+1; M:=M+1; END;
  T := GETT(ALONG, RANK);
  FACTOR := 1; TOP := RIGHT+ALONG;
  FOR N:=RIGHT+RANK-1 STEP -1 UNTIL TOP DO FACTOR:=
            FACTOR × SP[NOC];
  N:=RIGHT + RANK - 1; DIM := SP[NOC];
  N := N+1; M:=S+RANK; I:=0;
    DIMMOD := DIM-1;
    WHILE I LSS RSIZE DO BEGIN
      CASE T OF BEGIN
        L := I DIV FACTOR MOD LSIZE;
         L := I DIV FACTOR MOD DIMMOD;
         L := I MOD DIM; END;
      L := L+LEFT;
      IF SP[LOC] = 1 THEN FOR K:=1 STEP 1 UNTIL FACTOR DO BEGIN
       SP[MOC]:=SP[NOC]; I:=I+1; M:=M+1; N:=N+1;
        END ELSE BEGIN I:=I+FACTOR; N:=N+FACTOR; END;
      END;
    GO TO QUIT;
  RANKE:  ERR:=RANKERROR; GO TO QUIT;
  DOMAIN:  ERR:=DOMAINERROR; GO TO QUIT;
  QUIT:  IF CHARACTER THEN BEGIN PACK(S,RANK,SIZE);
            DESC.ARRAYTYPE:=1; FORGETSPACE(RIGHT,RSIZE+RANK); END;
    RESULTD := DESC;
    POP;
  END PROCEDURE COMPRESS;
PROCEDURE EXPAND(LDESC,RDESC,DIM); VALUE LDESC,RDESC,DIM;
    REAL LDESC, RDESC, DIM;
  BEGIN INTEGER I,J,K,L,M,N,S,T,RANK,LSIZE,RSIZE,SIZE,
              ALONG,TOP,LADDR,MADDR,FACTOR, SUM;
    REAL DESC, INSERT;
    LABEL QUIT, DOMAIN;
    BOOLEAN CHARACTER;
    LSIZE:=FINDSIZE(LDESC); RSIZE:=FINDSIZE(RDESC);
    RANK := RDESC.RF;
    IF M:=RDESC.SPF=0
      OR L:=LDESC.SPF=0
      OR I:=LDESC.RF GTR 1

      OR N:=DIM.SPF=0 AND DIM.DID NEQ 0
      OR DIM.ARRAYTYPE=1
      OR FINDSIZE(DIM ) NEQ 1
      OR LDESC.ARRAYTYPE=1
```

```
        THEN GO TO DOMAIN;
    N:=N + (T:=DIM.RF);
  IF ALONG :=(IF N=T THEN RANK ELSE SP[NOC]-ORIGIN+1) GTR RANK
          OR ALONG LSS 1 AND RANK NEQ 0
     THEN BEGIN ERR:=INDEXERROR; GO TO QUIT; END;
   IF RANK=0 THEN DIM:=1
       ELSE BEGIN N:=M+ALONG-1; DIM:=SP[NOC]; END;
  IF SIZE:=RSIZE DIV DIM × LSIZE GTR MAXWORDSTORE
        THEN BEGIN ERR:=KITEERROR; GO TO QUIT; END;
    IF NOT CHECKANDADD(LSIZE,LADDR:=L+T, SUM) THEN GO TO DOMAIN;
    IF SUM NEQ DIM THEN BEGIN ERR:=RANKERROR; GO TO QUIT; END;
  IF RANK=0 THEN BEGIN
    DIM:=SP[MOC]; DESC.SPF:=N:=GETSPACE(LSIZE+I);
    DESC.RF:=I; DESC.DID:=(IF I=0 THEN DDPUSW ELSE DDPUVW);
    SPCOPY(L,N,I); L:=L+I; N:=N+I; TOP:=L+LSIZE-1;
    FOR L:=L STEP 1 UNTIL TOP DO BEGIN
     IF SP[LOC]=1 THEN SP[NOC]:=DIM;
     N:=N+1; END;
  GO TO QUIT END;
    IF RDESC.ARRAYTYPE=1 THEN BEGIN CHARACTER:=TRUE;
        M:=UNPACK(M,RANK,RSIZE);
        INSERT := " "; END;
    FACTOR:=1; TOP:=M+ALONG;
    FOR N:=M+RANK-1 STEP -1 UNTIL TOP DO FACTOR:=FACTOR×SP[NOC];
  T := GETT(ALONG, RANK);
    J:=0; N:=(MADDR:=M) + RANK;
    DESC.SPF:=M:=GETSPACE(SIZE+RANK);
    I:=M+RANK;
    WHILE J LSS SIZE DO BEGIN
      CASE T OF BEGIN
        S := J DIV FACTOR MOD LSIZE;
         S:=J DIV FACTOR MOD LSIZE;
         S:=J MOD LSIZE; END;
    L:=S + LADDR;
    IF SP[LOC]=1  THEN FOR K:=1 STEP 1 UNTIL FACTOR DO
        BEGIN L:=J+I; SP[LOC] := SP[NOC];
          J:=J+1; N:=N+1;
        END ELSE FOR K:=1 STEP 1 UNTIL FACTOR DO BEGIN
          L:=J+I; SP[LOC]:=INSERT; J:=J+1; END;
     END;
    L := MADDR;
    FOR I:=1 STEP 1 UNTIL RANK DO BEGIN
      IF I = ALONG THEN SP[MOC]:=LSIZE ELSE SP[MOC]:=SP[LOC];
     M:=M+1; L:=L+1; END;
    DESC.DID:=DDPUVW; DESC.RF:=RANK;
    GO TO QUIT;
   DOMAIN:  ERR:=DOMAINERROR;
   QUIT:  IF CHARACTER THEN BEGIN DESC.ARRAYTYPE:=1;
                FORGETSPACE(MADDR, RSIZE+RANK);
                PACK(DESC.SPF,RANK,SIZE); END;
    RESULTD:=DESC;
    POP;
   END PROCEDURE EXPAND;
PROCEDURE MEMBER;
  BEGIN REAL LDESC, RDESC;
    INTEGER L,M,N,I,S,T,LSIZE,RSIZE,LRANK,RRANK,TOP;
    REAL DESC, TEMP, ANS;
    LABEL QUIT;
  LDESC := AREG; RDESC := BREG;
  LSIZE:=FINDSIZE(LDESC); RSIZE:=FINDSIZE(RDESC);
  LRANK:=LDESC.RF; RRANK:=RDESC.RF;
  IF L:=LDESC.SPF=0 OR M:=RDESC.SPF=0 THEN BEGIN
      ERR:=DOMAINERROR; GO TO QUIT END;
  IF LDESC.ARRAYTYPE=1 THEN L:=UNPACK(L,LRANK,LSIZE);
  IF RDESC.ARRAYTYPE=1 THEN M:=UNPACK(M,RRANK,RSIZE);
  DESC:=LDESC; DESC.NAMED:=0;
  DESC.ARRAYTYPE:=0;
  DESC.SPF:=N:=GETSPACE(LSIZE+LRANK);
  SPCOPY(L,N,LRANK);
  N:=N+LRANK; L:=(I:=L)+LRANK; M:=(S:=M)+RRANK;
  T:=M+RSIZE-1; TOP := L+LSIZE-1;
  FOR L:=L STEP 1 UNTIL TOP DO BEGIN
    TEMP:=SP[LOC]; M:=S;
    WHILE M LEQ T DO
      IF OPERATION(TEMP,SP[MOC],0,12,ANS) AND ANS=1 THEN BEGIN
        SP[NOC]:=1; M:=M+T; END ELSE M:=M+1;
   N:=N+1;  END;

  IF RDESC.ARRAYTYPE=1 THEN FORGETSPACE(S,RSIZE+RRANK);
  IF LDESC.ARRAYTYPE=1 THEN FORGETSPACE(I,LSIZE+LRANK);
  QUIT:  RESULTD:=DESC;
  END PROCEDURE MEMBER;
REAL PROCEDURE BASEVALUE;
```

```
    BEGIN
    COMMENT THIS RETURNS A DESCRIPTOR FOR A SCALAR RESULT;
    LABEL OUTE,BAD;
     REAL E,L,M,LEFT,RIGHT,T,LARG,RARG;
     LARG := AREG; RARG := BREG;
    IF M:=RARG.SPF=0 OR LARG.CHRMODE=1 OR RARG.CHRMODE=1
        OR L:=LARG.SPF=0 AND LARG.DID NEQ 0
            THEN GO TO BAD;
    RIGHT:=SP[MOC];
    LEFT:=SP[LOC];
    IF FINDSIZE(LARG)=1 THEN % A 1 ELEMENT VECTOR
        BEGIN
        L:=L+LARG.RF;
        LARG.SCALAR:=1;
        LEFT:=SP[LOC];
        END;
    IF FINDSIZE(RARG)=1 THEN % A ONE ELEMENT VECTOR
        BEGIN
        M:=M+RARG.RF;
        RIGHT:=SP[MOC];
        RARG.SCALAR:=1;
        END;
    IF L=0 THEN
        BEGIN % BASEVAL MONADIC
        LEFT:=2; %IF MONADIC, ITS 2 BASVAL X
        LARG.SCALAR:=1;
        END;
    IF BOOLEAN(LARG.SCALAR )THEN %SCALAR
        IF BOOLEAN(RARG.SCALAR) THEN
            BEGIN
            T:=RIGHT; %SCALAR-SCALAR
            GO OUTE;
            END
        ELSE
    IF RARG.RF=1 THEN
            BEGIN COMMENT SCALAR-VECTOR--LEFT IS VALUE OF SCALAR, RIGHT
                IS # OF ELEMENTS;
            IF LEFT=0 THEN GO OUTE
            ELSE E:=1/LEFT;
            FOR L :=M+RIGHT STEP -1 UNTIL M+1 DO
                T:=T+SP[LOC]×(E:=E×LEFT);
            GO OUTE;
            END
        ELSE BAD: ERR:=DOMAINERROR
    ELSE
    IF RARG.SCALAR=0 THEN
            IF LARG.RF NEQ 1 OR RARG.RF NEQ 1 THEN
                ERR:=DOMAINERROR
            ELSE
        BEGIN
        GT2:=L; % SAVE FOR LATER TEST
        GT1:=M+2; % WANT TO STOP 2 UP IN LOOP
        L:=L+LEFT; % START AT OTHER END
        E:=1;
        M:=M+RIGHT;
        T:=SP[MOC]; % INITIAL VALUE
        FOR M:=M-1 STEP -1 UNTIL GT1 DO
            BEGIN
        IF L:=L-1 LSS GT2 THEN L:=GT2+LEFT; % START OVER
            E:=E×SP[LOC];
            T:=T+SP[MOC]×E;
            END;
OUTE:
        L:=GETSPACE(1);
        SP[LOC]:=T;
        T:=0;
        T.DID:=DDPUSW; % BUILD DESCRIPTOR
        T.SPF:=L;
        BASEVALUE:=T;
        END
        ELSE ERR := DOMAINERROR
    END OF BASEVALUE;
REAL PROCEDURE REPRESENT;
    BEGIN
    COMMENT RETURNS DESCRIPTOR OF VECTOR IF LARG VECTOR AND RARG SCALAR;
     REAL L,M,LEFT,RIGHT,T,E,LARG,RARG;
    LABEL AROUND;
     LARG := AREG; RARG := BREG;
    IF (RARG.SCALAR=1 OR FINDSIZE(RARG)=1 AND RARG.CHRMODE=0)
        AND NOT(LARG.SCALAR=1 OR LARG.CHRMODE=1 OR LARG.RF NEQ 1) THEN
        BEGIN
        COMMENT VECTOR-SCALAR;
        IF L:=LARG.SPF=0 OR M:=RARG.SPF=0 THEN GO AROUND;
IF MARG.SCALAR=0 THEN M:=M+MARG.RF;
        RIGHT:=SP[MOC]; % VALUE OF SCALAR
        LEFT:=SP[LOC]; % LENGTH OF VECTOR
        E:=M:=GETSPACE(LEFT+1); % MAKE ROOM FOR ANSWER
        SP[MOC]:=LEFT; % LENGTH OF ANSWER
        M:=M+LEFT;
        GT1:=L+2;
        FOR L:=L+LEFT STEP -1 UNTIL GT1 DO
            IF T:=SP[LOC] LEQ 0 THEN
                IF T LSS 0 THEN ERR := DOMAINERROR
               ELSE
                   BEGIN
                   L:=GT1-1 ; % STOP THE LOOP
                   M:=M-1;
                   END
           ELSE
           BEGIN
           SP[MOC]:= RIGHT MOD T;
           RIGHT:=RIGHT DIV T;
           M:=M-1;
           IF RIGHT LSS FUZZ THEN L:=GT1-1; % STOP THE LOOP
           END;
        SP[MOC]:=RIGHT; % LEFTOVER GOES HERE
        T.DID:=DDPUVW;
        T.RF:=1;
        T.SPF:=E;
        REPRESENT:=T;
        END
    ELSE AROUND: ERR:=DOMAINERROR;
    END OF REPRESENT;
PROCEDURE PERIOD(LDESC,RDESC,LOP,ROP);
   VALUE LDESC,RDESC,LOP,ROP; REAL LDESC,RDESC; INTEGER LOP,ROP;
BEGIN INTEGER L,M,N,J,LRANK,RRANK,RANK,LSIZE,RSIZE,SIZE,LL,MM,I,
              RROW,RCOL,LROW,LCOL,LJUMP,RJUMP,MSAVE,LSAVE,RSTART;
   REAL DESC, TEMP;
   BOOLEAN CHARACTER, FIRST,LSCALAR, RSCALAR;
   LABEL QUIT, DOMAIN, FORGET, OUTERPROD;
   IF L:=LDESC.SPF = 0 OR M := RDESC.SPF=0 THEN GO TO DOMAIN;
   LSIZE := FINDSIZE(LDESC); RSIZE:=FINDSIZE(RDESC);
   LRANK:=LDESC.RF;  RRANK := RDESC.RF;
   IF LOP NEQ 45 THEN
   IF LRANK GTR 2 AND LSIZE NEQ 1 OR RRANK GTR 2 AND RSIZE NEQ 1 THEN
    BEGIN ERR:=KITEERROR; GO TO QUIT; END;
   IF ROP:=GETOP(CORRESPONDENCE,ROP-1) = 9 THEN BEGIN
       ERR:=SYNTAXERROR; GO TO QUIT; END;
   IF LL:=LDESC.ARRAYTYPE=1 OR MM:=RDESC.ARRAYTYPE=1 THEN
      IF LL × MM NEQ 1 THEN GO TO DOMAIN
    ELSE BEGIN

      IF ROP LSS 11 OR ROP GTR 16 THEN GO TO DOMAIN;
      CHARACTER:=TRUE;
      M:=UNPACK(M,RRANK,RSIZE);
      L:=UNPACK(L,LRANK,LSIZE); END;
  MSAVE := M; LSAVE:=L;  IF ROP NEQ 45 THEN
  IF LOP=45 THEN GO TO OUTERPROD ELSE
    IF LOP:=GETOP(CORRESPONDENCE,LOP-1)=9 THEN
         BEGIN ERR:=SYNTAXERROR; GO TO QUIT; END;
  IF  LRANK=2 THEN BEGIN
      N:=L+LRANK-1; LCOL := SP[NOC];
      N:=N-1; LROW:=SP[NOC]; END;
  IF  LRANK=1 THEN BEGIN LROW:=1; LCOL:=SP[LOC]; END;
  IF  RRANK=2 THEN BEGIN
      N :=M+RRANK-1; RCOL:=SP[NOC];
      N:=N-1; RROW:=SP[NOC]; END;
  IF  RRANK=1 THEN BEGIN RROW:=SP[MOC]; RCOL:=1; END;
  IF  LSIZE =1 OR RSIZE=1 THEN BEGIN
  IF  LSIZE = 1 AND RSIZE = 1 THEN LROW:=LCOL:=RROW:=RCOL:=1
      ELSE IF LSIZE=1 THEN BEGIN LCOL:=RROW; LROW:=1;
                             L:=L+LRANK-1; LRANK:=1;
                             LSCALAR:=TRUE; END
      ELSE BEGIN RROW := LCOL; RCOL := 1;
                              M:=M+RRANK-1; RRANK:=1;
                              RSCALAR:=TRUE; END;
  END;
  IF LCOL NEQ RROW
     THEN BEGIN ERR:=RANKERROR; GO TO QUIT; END;
  DESC.SPF:=N:=GETSPACE((RANK:=MAX(0,LRANK+RRANK-2))+
                         SIZE:=LROW×RCOL);
  SPCOPY(L,N,LRANK-1);
  SPCOPY(M+1,N+LRANK-1,RRANK-1);
  DESC.RF:=RANK; DESC.DID:=(IF RANK=0 THEN DDPUSW ELSE DDPUVW);
  N:=N+RANK;
  LL := L + LRANK - 1;
```

```
    MM := M + RRANK - 1;
    LJUMP := LCOL-1; RJUMP := IF RSCALAR THEN 0 ELSE (RROW-1) × RCOL;
    FOR J:=1 STEP LCOL UNTIL LSIZE DO
      FOR RSTART:=1 STEP 1 UNTIL RCOL DO BEGIN
        FIRST:=TRUE;
        M := MM + RSTART + RJUMP; RROW := LL+J;
        FOR I:=LL + LJUMP + J STEP -1 UNTIL RROW DO BEGIN
          IF LSCALAR THEN L:=LL+1 ELSE L:=I;
          IF FIRST THEN BEGIN
            IF NOT OPERATION(SP[LOC],SP[MOC],1,ROP,SP[NOC])
               THEN GO TO FORGET ELSE FIRST := FALSE;
          END ELSE BEGIN
            IF NOT OPERATION(SP[LOC],SP[MOC],1,ROP,TEMP)
               THEN GO TO FORGET;
            IF NOT OPERATION(TEMP,SP[NOC],-1,LOP,SP[NOC])
               THEN GO TO FORGET; END;
          IF NOT RSCALAR THEN M:=M-RCOL; END;
        N := N+1;
        END;
   GO TO QUIT;
 OUTERPROD:  IF SIZE:=LSIZE×RSIZE GTR MAXWORDSTORE
     OR RANK := LRANK+RRANK GTR 31 THEN BEGIN
      ERR:=KITEERROR; GO TO QUIT; END;
    DESC.SPF:=N:=GETSPACE(SIZE+RANK);
    DESC.DID:=IF RANK=0 THEN DDPUSW ELSE DDPUVW;
    DESC.RF:=RANK;
    SPCOPY(L,N,LRANK);
   SPCOPY(M,N+LRANK,RRANK);
    N:=N+RANK;
    I:=L + LRANK + LSIZE - 1;
    MM := M+RRANK + RSIZE - 1;
    FOR L:=L+LRANK STEP 1 UNTIL I DO
      FOR M:=MSAVE+RRANK STEP 1 UNTIL MM DO
       IF NOT OPERATION(SP[LOC],SP[MOC],1,ROP,SP[NOC]) THEN
          GO TO FORGET ELSE N:=N+1;
   GO TO QUIT;
   FORGET:  FORGETSPACE(DESC.SPF,RANK+SIZE);
   DOMAIN:  ERR:=DOMAINERROR;
   QUIT:  IF CHARACTER THEN BEGIN
          FORGETSPACE(MSAVE , RRANK+RSIZE);
          FORGETSPACE(LSAVE , LRANK+LSIZE); END;
    RESULTD := DESC;
  END PROCEDURE PERIOD;
 PROCEDURE REVERSE(SOURCE,LENGTH,DEST,JUMP); VALUE SOURCE,DEST,
      LENGTH,JUMP; INTEGER SOURCE,LENGTH,DEST,JUMP;
   BEGIN INTEGER L,M,TOP;
     M:=SOURCE + TOP:=(LENGTH-1) × JUMP; TOP:=DEST+TOP;
     FOR L:=DEST STEP JUMP UNTIL TOP DO BEGIN
       SP[LOC] := SP[MOC]; M:=M-JUMP; END;
   END PROCEDURE REVERSE;
 PROCEDURE ROTATE(SOURCE,LENGTH,DEST,JUMP,ROT); VALUE SOURCE,
   LENGTH,DEST,JUMP,ROT; INTEGER SOURCE,LENGTH,DEST,JUMP,ROT;
   BEGIN INTEGER L,M,TOP;
     TOP := SOURCE + (LENGTH-1) × JUMP;
     FOR L:=SOURCE STEP JUMP UNTIL TOP DO BEGIN
       M:=DEST+(ROT MOD LENGTH)×JUMP; SP[MOC]:=SP[LOC];
       ROT := ROT + 1; END;
   END PROCEDURE ROTATE;
 INTEGER PROCEDURE GETNUM(TIM,L,SIZE,DIM); VALUE TIM,L,
      SIZE,DIM; INTEGER TIM,L,SIZE,DIM;
   BEGIN INTEGER NUM;
     IF SIZE NEQ 0 THEN L := L + TIM;
       NUM:=SIGN(NUM:=SP[LOC]) × ENTIER(ABS(NUM)) MOD DIM;
       IF NUM LSS 0 THEN GETNUM := -NUM  %FOR RIGHT ROTATION
                    ELSE GETNUM:=DIM-NUM; %FOR LEFT ROTATION
   END PROCEDURE GETNUM;
 BOOLEAN PROCEDURE MATCHROT(LDESC,RDESC,ALONG); VALUE LDESC,
      RDESC,ALONG; INTEGER LDESC,RDESC,ALONG;
   BEGIN INTEGER I,L,M,R; LABEL QUIT;
     MATCHROT:=TRUE; L:=LDESC.SPF; M:=RDESC.SPF;
     IF R:=LDESC.RF NEQ RDESC.RF-1 THEN BEGIN
          MATCHROT:=FALSE; GO TO QUIT; END;
    FOR I:=1 STEP 1 UNTIL R DO BEGIN IF I=ALONG THEN M:=M+1;
       IF SP[LOC] NEQ SP[MOC] THEN BEGIN MATCHROT:=FALSE;
             GO TO QUIT; END; M:=M+1; L:=L+1; END;
   QUIT:  END PROCEDURE MATCHROT;
 PROCEDURE REDUCESORTSCAN(LOP,RDESC,DIM,KIND); VALUE LOP,RDESC,
   DIM,KIND; REAL LOP,RDESC,DIM; INTEGER KIND;
   BEGIN INTEGER L,M,N,I,J,K,ALONG,FACTOR,T,MSAVE,DIFF,SSIZE,
              JUMP,RANK,SIZE,TOP,LASTDIM,INTERVAL,TEMP,HOP;
     INTEGER REMDIM,LRANK,LSAVE,LSIZE,S;
     BOOLEAN CHARACTER,REDUCE,SORT,SCAN,REVERSAL,ROTATION;
     REAL DESC;
    LABEL QUIT, FORGET, RANKERR;
  COMMENT:   KIND=1 FOR REDUCTION
             KIND=2 FOR SORTUP OR SORTDN
             KIND=3 FOR SCAN
             KIND=4 FOR REVERSAL
             KIND=5 FOR ROTATION;
   PROCEDURE SORTIT(L,M,SIZE,JUMP,UP); VALUE L,M,SIZE,JUMP,UP;
         INTEGER L,M,SIZE,JUMP; BOOLEAN UP;
     BEGIN INTEGER N,TIP,TOP,LSAVE;
     REAL COMPARE,OUTOFIT;
     OUTOFIT:=IF UP THEN BIGGEST ELSE -BIGGEST;
     TIP := M + (N:=(SIZE-1)) × JUMP; TOP := L + N;
     LSAVE := L;
     FOR M:=M STEP JUMP UNTIL TIP DO BEGIN
       L := LSAVE; COMPARE := SP[LOC]; N:=L;
       FOR L:=L+1 STEP 1 UNTIL TOP DO
       IF UP THEN BEGIN IF SP[LOC] LSS COMPARE THEN BEGIN
              N:=L; COMPARE:=SP[LOC]; END;
        END ELSE IF SP[LOC] GTR COMPARE THEN BEGIN
              N:=L; COMPARE:=SP[LOC]; END;
       SP[NOC] := OUTOFIT;
       SP[MOC] := (N-LSAVE) + ORIGIN;
    END;
   END PROCEDURE SORTIT;
 CASE KIND OF BEGIN ; REDUCE:=TRUE; SORT:=TRUE; SCAN:=TRUE;
            REVERSAL:=TRUE; ROTATION:=TRUE; END;
 IF LOP GTR 64 AND NOT ROTATION THEN BEGIN
       ERR:=SYSTEMERROR; GO TO QUIT; END;
 IF REDUCE OR SCAN THEN IF LOP NEQ 45 THEN
       LOP := GETOP(CORRESPONDENCE,LOP-1);
 IF M:=RDESC.SPF=0 AND NOT REDUCE
   OR DIM.DID NEQ 0 AND N:=DIM.SPF=0 OR DIM.ARRAYTYPE=1
   OR FINDSIZE(DIM) NEQ 1 THEN BEGIN
      ERR:=DOMAINERROR; GO TO QUIT END;
   IF (REDUCE OR SCAN) AND LOP=9 THEN BEGIN %OP NOT DYADIC SCALAR
      ERR:=SYNTAXERROR; GO TO QUIT END;
  IF M=0 THEN BEGIN
    %FOR REDUCTION, RESULT OF A NULL IS CORRESPONDING IDENTITY
   %EXCEPT THAT NAND, NOR, CIRCLE, AND LOG (LOP GTR 18)
    %HAVE NO IDENTITIES, SO THE RESULT IS A NULL
  DESC.DID := DDPUSW;
  IF LOP LEQ 18 THEN BEGIN DESC.SPF:=N:=GETSPACE(1);
    SP[NOC] := IDENTITY(LOP); END ELSE DESC.RF:=1;
  GO TO QUIT; END;
 IF RDESC.ARRAYTYPE=1 AND (REDUCE OR SCAN) THEN
     BEGIN ERR:=DOMAINERROR; GO TO QUIT; END;
 SIZE:=FINDSIZE(RDESC);
 RANK:=RDESC.RF;
 IF SIZE=1 THEN BEGIN
  %UNLESS SORT, RESULT OF SINGLE-VALUED ARGUMENT IS THAT ARGUMENT
   DESC := RDESC;
   DESC.SPF := N := GETSPACE(RANK+1);
      SPCOPY(M,N,RANK);M:=M+RANK;N:=N+RANK;
   IF SORT THEN BEGIN SP[NOC]:=ORIGIN; DESC.ARRAYTYPE:=0;
         END ELSE SP[NOC]:=SP[MOC];
   GO TO QUIT; END;

 IF RDESC.ARRAYTYPE=1 THEN BEGIN
   CHARACTER := TRUE;
   M:=UNPACK(M,RANK,SIZE); END;
 MSAVE:=M;
 N:=N+(T:=DIM.RF);
 IF ALONG:=(IF N=T THEN RANK ELSE SP[NOC]-ORIGIN+1) GTR RANK
   OR ALONG LSS 1
   THEN BEGIN ERR:=INDEXERROR; GO TO QUIT; END;
IF ROTATION THEN BEGIN
   IF LSAVE:=LOP.SPF=0 OR LOP.ARRAYTYPE=1 THEN
       BEGIN ERR:=DOMAINERROR; GO TO QUIT; END;
   IF LSIZE:=FINDSIZE(LOP) NEQ 1 THEN
       IF NOT MATCHROT(LOP,RDESC,ALONG) THEN BEGIN
           ERR:=RANKERROR; GO TO QUIT; END;
   LSAVE := LSAVE + LRANK := LOP.RF;
   IF LSIZE = 1 THEN LRANK := 0; END;
 N:=M+ALONG-1;
 DIM:=SP[NOC];
 JUMP:=1; I:=M+ALONG;
 FOR L:=M+RANK-1 STEP -1 UNTIL I DO JUMP:=JUMP × SP[LOC];
 N:=M+RANK-1; LASTDIM:=SP[NOC];
 IF ALONG = RANK-1 THEN BEGIN N:=N-1;
   FACTOR:=LASTDIM × SP[NOC]; END;
 T := GETT(ALONG, RANK);
 J := M + RANK;
 REMDIM := 1;
```

```
    HOP := (DIM-1) × JUMP;
    DESC.DID := DDPUVW;
    IF ALONG GTR 1 AND ALONG LSS RANK-1 THEN BEGIN TOP:=M+ALONG-2;
      FOR L:=M STEP 1 UNTIL TOP DO REMDIM:=REMDIM×SP[LOC]; END;
    IF REDUCE THEN BEGIN DESC.SPF:=N:=GETSPACE(SSIZE:=SIZE DIV DIM
                                             + RANK - 1);
       IF RANK=1 THEN DESC.SCALAR:=1 ELSE DESC.RF:=RANK-1;
       FOR I:=1 STEP 1 UNTIL RANK DO BEGIN
        IF I NEQ ALONG THEN BEGIN SP[NOC]:=SP[MOC]; N:=N+1; END;
         M:=M+1; END;
       JUMP := - JUMP;
    END ELSE BEGIN DESC.SPF:=N:=GETSPACE(SSIZE:=SIZE+RANK);
       INTERVAL := (DIFF := N-M) + HOP;
       SPCOPY(M,N,RANK); DESC.RF:=RANK; END;
    IF SORT THEN TEMP:= GETSPACE(DIM);
    TOP := SIZE DIV (DIM × REMDIM) - 1;
    FOR S:=1 STEP 1 UNTIL REMDIM DO BEGIN
    FOR I:=0 STEP 1 UNTIL TOP DO BEGIN
      CASE T OF BEGIN
        L := I + J;
        L:=I DIV LASTDIM×FACTOR + I MOD LASTDIM + J;
        L:=I×LASTDIM + J; END;
      IF REDUCE THEN BEGIN M:=I+N;  L:=HOP + (K:=L);
        SP[MOC] := SP[LOC];
        FOR L:=L+JUMP STEP JUMP UNTIL K DO
           IF NOT OPERATION(SP[LOC],SP[MOC],-1,LOP,SP[MOC])
               THEN GO TO FORGET;
       END ELSE
      IF SORT THEN BEGIN K:=L+HOP; N:=TEMP;
         FOR M:=L STEP JUMP UNTIL K DO BEGIN
          SP[NOC] := SP[MOC]; N:=N+1; END;
         IF LOP LSS 0 THEN SORTIT(TEMP,L+DIFF,DIM,JUMP,FALSE)
                       ELSE SORTIT(TEMP,L+DIFF,DIM,JUMP,TRUE);
      END ELSE IF SCAN THEN BEGIN
       K:=L+INTERVAL; N:=L+DIFF; SP[NOC] := SP[LOC];
       FOR N:=N+JUMP STEP JUMP UNTIL K DO BEGIN
           M:=N-JUMP; L:=L+JUMP;
           IF NOT OPERATION(SP[MOC],SP[LOC],-1,LOP,SP[NOC])
             THEN GO TO FORGET; END;
    END ELSE IF REVERSAL THEN REVERSE(L,DIM,L+DIFF,JUMP)
    ELSE IF ROTATION THEN ROTATE(L,DIM,L+DIFF,JUMP,
                    GETNUM(I,LSAVE,LRANK,DIM));
      END;
    J := J + ABS(JUMP×DIM);
    N := N + TOP + 1;
    DIFF := DIFF + TOP + 1;
    END;
    GO TO QUIT;
    RANKERR:  ERR:=RANKERROR; FORGETSPACE(DESC.SPF,SSIZE);GO QUIT;
    FORGET:  ERR:=DOMAINERROR; FORGETSPACE(DESC.SPF, SSIZE);
  QUIT:  IF CHARACTER THEN BEGIN
       FORGETSPACE(MSAVE,SIZE+RANK);
       IF (REVERSAL OR ROTATION) AND ERR=0 THEN BEGIN
         DESC.ARRAYTYPE:=1; PACK(DESC.SPF,RANK,SIZE); END; END;
      IF SORT THEN FORGETSPACE(TEMP,DIM);
    RESULTD := DESC;
     IF ROTATION THEN POP;
   END PROCEDURE REDUCESORTSCAN;
  PROCEDURE DYADICTRANS;
  BEGIN REAL LDESC,RDESC;
    INTEGER L,M,N,RANK,NEWRANK,SIZE,TEMP,I,J;
    DEFINE SPTOP=RDESC#,MIN=RDESC#,PTR=NEWRANK#,MBASE=LDESC#,TOP=RDESC#
          ,RESULT=RESULTD#;
    LABEL QUIT; BOOLEAN CARRY;
  INTEGER ARRAY RVEC,DEL,SUB,OLDEL[0:31];
    LDESC:=AREG; RDESC:=BREG;
    RESULT:=0; L:=LDESC.SPF; J:=LDESC.RF; RANK:=RDESC.RF;
    IF M:=RDESC.SPF=0 OR L=0 OR LDESC.ARRAYTYPE=1 THEN BEGIN
          ERR:=DOMAINERROR; GO TO QUIT END;
    IF NUMELEMENTS(LDESC)=1 THEN BEGIN N:=L+J;
      IF SP[NOC] NEQ ORIGIN OR RANK GTR 1 THEN BEGIN
           ERR:=DOMAINERROR; GO TO QUIT END;
     %IF WE GET HERE, THE ANSWER IS ITSELF
      RESULT:=RDESC; I:=NUMELEMENTS(RDESC);
      RESULT.SPF:=N:=GETSPACE(SIZE:=RANK+I); RESULT.NAMED:=0;
      SPCOPY(M,N,SIZE); GO TO QUIT; END;
    IF J GTR 1 THEN BEGIN ERR:=RANKERROR; GO TO QUIT END;
    IF SP[LOC] NEQ RANK THEN BEGIN ERR:=LENGTHERROR; GO TO QUIT END;
  % FIND MAX OF LDESC FOR NOW- DO THE REST LATER
  %LDESC W/R/T/ ORIGIN 0 GETS STORED IN SUB[I]
    SPTOP:=L+RANK; NEWRANK:=0; I:=0;
    FOR N:=L+1 STEP 1 UNTIL SPTOP DO BEGIN
      IF TEMP:=SP[NOC]-ORIGIN+1 GTR NEWRANK THEN NEWRANK:=TEMP;
      SUB[I]:=TEMP-1; I:=I+1 END;
    IF NEWRANK GTR RANK THEN BEGIN ERR:=DOMAINERROR;GO TO QUIT END;
  % CALCULATE THE OLD DEL VECTOR, OLDEL
    OLDEL[RANK-1]:=1; N:=M+RANK-1;
    FOR I:=RANK-2 STEP -1 UNTIL 0 DO BEGIN
        OLDEL[I]:=OLDEL[I+1]×SP[NOC]; N:=N-1 END;
    MBASE:=M; SIZE:=1;
  %FIX UP THE NEW RVEC AND DEL
    FOR I:=NEWRANK-1 STEP -1 UNTIL 0 DO BEGIN
  % FIND SMALLEST EL. OF RHO RDESC [J] S.T. A[J]=I
  %          AND SUM OF OLDEL[J] S.T. A[J]=I
    MIN:=31; TEMP:=0;
    FOR J:=RANK-1 STEP -1 UNTIL 0 DO
        IF SUB[J]=I THEN BEGIN
          M:=MBASE+J;
          IF SP[MOC] LSS MIN THEN MIN:=SP[MOC];
          TEMP:=TEMP+OLDEL[J] END;
        RVEC[I]:=MIN; DEL[I]:=TEMP; SIZE:=SIZE×RVEC[I];
        IF TEMP=0 THEN BEGIN %IT DOESN7T EXHAUSE IOTA NEWRANK
            ERR:=DOMAINERROR; GO TO QUIT END;
      END;
    RESULT:=M:=GETSPACE(NEWRANK+SIZE);
    RESULT.RF:=NEWRANK; RESULT.DID:=DDPUVW;
    IF BOOLEAN(BREG.ARRAYTYPE) THEN BEGIN
        RESULT.ARRAYTYPE:=1; N:=MBASE;
        MBASE:=UNPACK(MBASE,RANK,N:=OLDEL[0]×SP[NOC]);
        FORGETSPACE(MBASE,N+RANK) END;
    FOR I:=1 STEP 1 UNTIL NEWRANK DO BEGIN
      SP[MOC]:=RVEC[I-1]; M:=M+1 END;
   %INITIALIZE FOR STEPPING THRU NEW ARRAY
    FOR I:=NEWRANK-1 STEP -1 UNTIL 0 DO BEGIN
        SUB[I]:=0; OLDEL[I]:=RVEC[I]×DEL[I] END;
    L:=MBASE+RANK;
  %STEP THRU THE SUBSCRIPTS OF THE ANSWER TO PICK UP THE ELEMENTS
  %     IN ROW ORDER ACCORDING TO THE MAPPING GIVEN BY DEL
    PTR:=TOP:=NEWRANK-1;
    FOR I:=1 STEP 1 UNTIL SIZE DO BEGIN
        SP[MOC] :=SP[LOC];
        M:=M+1;
  %GET NEXT SUBSCRIPT FOR NEW ARRAY AND SET NEXT L;
        SUB[PTR]:=SUB[PTR]+1;
        L:=L+DEL[TOP];
        CARRY:=TRUE;
        WHILE CARRY AND I NEQ SIZE DO
          IF SUB[PTR] GEQ RVEC[PTR] THEN BEGIN
            SUB[PTR]:=0;
            L:=L-OLDEL[PTR]+DEL[PTR:=PTR-1];
            SUB[PTR]:=SUB[PTR]+1
            END ELSE CARRY := FALSE;
        PTR:=TOP;
        END;
    IF BOOLEAN(RESULT.ARRAYTYPE) THEN PACK(RESULT,SPF,TOP+1,SIZE);
  QUIT:  END OF DYADICTRANS;
      INTEGER PROCEDURE LOCATE(L,M); VALUE L,M; REAL L,M;
          BEGIN
          COMMENT L IS THE DIMENSION VECTOR(DESCRIPTOR),
          M IS THE INDEX VECTOR;
          INTEGER P,I,UB;
          L:=I:=L.SPF; M:=I:=M.SPF;
          UB:=SP[MOC]-1;
          M:=M+1;
          FOR I:=1 STEP 1 UNTIL UB DO
              BEGIN
              L:=L+1;
              P:=(P+SP[MOC]-1)×SP[LOC];
              M:=M+1
              END;
          P:=P+SP[MOC];
          LOCATE:=P+L;
          END;
      PROCEDURE DISPLAY(A,B); VALUE A,B; REAL A,B;
          BEGIN
      PROCEDURE PRINTMATRIX(L,ROW,COL);VALUE L,ROW,COL;
          INTEGER L,ROW,COL;
      BEGIN INTEGER I,J,CC,FOLD; DEFINE WIDE=GT2#;
      WIDE:=LINESIZE;
          FOR I:=1 STEP 1 UNTIL ROW DO
              BEGIN CC:=0; %NO BLANKS AT BEGINNING OF LINE
          FOLD:=0;
              FOR J:=1 STEP 1 UNTIL COL DO
                  BEGIN NUMBERCON(SP[LOC],ACCUM);
                  IF FOLD:=FOLD+ACOUNT+CC GTR WIDE AND ACOUNT+CC
                     LEQ WIDE THEN BEGIN TERPRINT;
```

```
             FORMROW(0,2,ACCUM,2,ACOUNT); FOLD:=ACOUNT+2; END ELSE
            FORMROW(0,CC,ACCUM,2,ACOUNT); L:=L+1;
            CC:=2; %PUT 2 BLANKS AFTER THE FIRST ITEM.
            END;
         TERPRINT;
         END
      END;
      INTEGER L,N,M,BOTTOM,ALOC,BLOC;
      INTEGER ROW,COL;
      ALOC:=A.SPF; BLOC:= B.SPF-1;
      L:=(M:=B.RF)+ BLOC; COL:=SP[LOC];
      L:=L-1;
      ROW:=(IF M GTR 1 THEN SP[LOC] ELSE 1);
      L:=BOTTOM:=M-2;
      PRINTMATRIX(LOCATE(B,A),ROW,COL);
      WHILE L GTR 0 DO
         BEGIN
         M:=ALOC+L; N:=BLOC+L;
         IF SP[MOC]:=SP[MOC]+1 GTR SP[NOC] THEN
            BEGIN SP[MOC]:=1; L:=L-1; END
            ELSE BEGIN FORMWD(3,"1         ");
            PRINTMATRIX(LOCATE(B,A),ROW,COL);
            L:=BOTTOM;
            END;
         END;
      FORMWD(3,"1         ");
       END;
   PROCEDURE MAKEFUNCTIONPRESENT(L); VALUE L ; REAL L; %LOC DESC
      BEGIN
      INTEGER I;
      REAL M,N,SEQ,ORD,D;
      BOOLEAN NUMERIC;
      REAL STREAM PROCEDURE CON(A); VALUE A;
         BEGIN SI:=LOC A; DI:=LOC CON; DS:=8OCT
         END;
      D:=SP[LOC]; %DESCRIPTOR FOR FUNCTION IS IN D
      SEQ:=GETFIELD(D,FSQF-8,FFL); ORD:=GETFIELD(D,FPTF-8,FFL);
      N:=GETSPACE((M:=SIZE(ORD))×2+6); %GET SPACE FOR TABLE
      SP[NOC]:=M×2+5; %SIZE OF THE VECTOR WHICH FOLLOWS
      D:=D&N[CSPF]&1[CRF]&0[BACKPT]; D.PRESENCE:=1;
      SP[LOC]:=D; %THIS SETS UP THE FUNCTION DESCRIPTOR.
      N:=N+1; SP[NOC]:=SEQ;
      COMMENT
         SP[N]   = SIZE OF THE VECTOR
         SP[N+1] = SEQUENTIAL STORAGE UNIT FOR THE TEXT
         SP[N+2] = SP LOC OF FIRST NUMERIC POINTER TO TEXT

         SP[N+3] = REL LOC (TO N+5) OF THE FIRST ARG
         SP[N+4] = REL LOC OF THE SECOND ARG
         SP[N+5] = REL LOC OF RESULT .  IF ANY ARE ZERO, THEN
                 THEY ARE NOT THERE.;
      D:=M; M:=(N:=N+4)+1; %D IS #ITEMS, M IS LOC 1ST, N=M-1
      FOR I:=1 STEP 1 UNTIL D DO %GET LABELS FROM STORAGE
         BEGIN L:=CONTENTS(ORD,I-1,GTA);
         IF NOT NUMERIC THEN %RESULT, ARGS, OR LOCALS/LABELS
            IF NUMERIC:=GTA[0]=0 THEN %FIRST NUMERIC POINTER
               BEGIN L:=N-3; SP[LOC]:=N+I×2-1;
               END;
         SP[MOC]:=GTA[0]; M:=M+1;
         IF NUMERIC THEN SP[MOC]:=GTA[1] ELSE
            BEGIN
            IF SEQ:=GTA[1] LSS 0 THEN %RESULT OR ARG
               BEGIN L:=N+SEQ+1; SP[LOC]:=I;
               SEQ:=0;
               END ELSE SEQ:=CON(SEQ)/10000;
            SP[MOC]:=SEQ
            END;
         M:=M+1
         END;
      COMMENT WE HAVE SET UP THE FUNCTION LABEL TABLE, LET
      SOMEONE ELSE FIGURE OUT HOW TO EXECUTE IT;
      END;
PROCEDURE PUSHINTOSYMTAB(FPTR);VALUE FPTR;REAL FPTR;
   BEGIN COMMENT ...PUT THE LOCAL VARIABLES FROM THIS SUSPENDED
   FUNCTION INTO THE SYMBOL TABLE TO BE TREATED AS GLOBAL VARIABLES
   WHILE THE FUNCTION IS SUSPENDED. FPTR IS THE ENTRY FROM THE
   STATE INDICATOR VECTOR FOR THE FUNCTION.;

   REAL T,U;
   LABEL COPY;
   INTEGER K,L,M,N;
   M:=FPTR.LOCFIELD+1;%LOCATE FMKS TO FIND LOCAL VALUES IN STACK
   N:=FPTR.SPF+2;T:=SP[NOC]-2;%FIND LOCAL NAMES
```

```
     FOR N:=N+4 STEP 2 UNTIL T DO %ONCE FOR EACH LOCAL
          BEGIN GT1:=SP[NOC].[6:42];%PICK UP THE LOCAL NAME
          L:=SYMBASE;K:=L+SP[LOC];% LOOK IN SYMBOL TABLE
          FOR L:=L+1 STEP 2 UNTIL K DO % CHECK EACH NAME
               IF GT1=SP[LOC].[6:42] THEN % WE FOUND A MATCH
               BEGIN GT1:=M;K:=M:=GETSPACE(1);L:=L+1;
               SP[MOC]:=SP[LOC]; %PUSH CURRENT DESCRIPTOR DOWN
               M:=GT1; GO TO COPY;
               END;
          COMMENT GET HERE IF NO MATCH...MUST MAKE A NEW ENTRY IN
          SYMBOL TABLE;
          IF K LSS MAXSYMBOL×2 THEN % THERE IS ROOM IN SYMBOL TABLE
               BEGIN L:=SYMBASE;SP[LOC]:=SP[LOC]+2; L:=K+1;
               SP[LOC]:=GT1&OPERAND[CTYPEF]&1[CSUSVAR];L:=L+1;K:=0;
COPY:          COMMENT L IS LOC IN SYMBOL TABLE FOR DESC. K WILL BE
               CONTENTS OF BACKF. NOW SET UP THE NEW DESCRIPTOR AND
               SAVE ITS LOCATION IN THE STACK. M IS THE STACK LOCATION
               OF THE LOCAL;

               SP[LOC]:=SP[MOC]&K[CLOCF]&1[CNAMED];
               SP[MOC]:=L&DDNUVW[CDID];M:=M+1;
               END ELSE % THERE IS NO ROOM IN THE SYMBOL TABLE
               BEGIN N:=T;ERR:=SPERROR;END;
          END;% OF FOR LOOP STEPPING THRU THE LOCALS
     END; % OF PUSHINTOSYMTAB PROCEDURE
PROCEDURE FORGETPROGRAM(U);VALUE U; REAL U;
     BEGIN REAL L,M;
     COMMENT U IS A PROGRAMMKS...THE SP STORAGE FOR THIS LINE
     SHOULD BE RELEASED;
     M:=U.SPF;SCRATCHAIN(SP[MOC].LOCFIELD);%CONSTANT CHAIN
     L:=SP[MOC].SPF;FORGETSPACE(M,1);%FORGET PROGRAM DESC.
     M:=L+1;SCRATCHDATA(SP[MOC]);%FORGET BUFFER
     FORGETSPACE(L,SP[LOC]+1);%FORGET THE POLISH
     END;
     EXPOVR:=EXPOVRL;
     INTOVR:=INTOVRL;
     INDEX:=INDEXL;
     FLAG:=FLAGL;
     ZERO:=ZEROL;
CASE MODE OF
BEGIN ;%-------------------------------------------------------------
%-------------- CASE 1....MODE=XEQUTE------------------------------
     CASE  CURRENTMODE OF
     BEGIN%---------------------------------------------------------
     %----------- SUB-CASE 0....CURRENTMODE=CALCMODE-----------
     IF T:=ANALYZE(TRUE) NEQ 0 THEN % WE HAVE A PROGRAM DESC
          BEGIN COMMENT SET-UP THE STACK;
     IF STACKBASE=0 THEN BEGIN
          STACKBASE:=L:=GETSPACE(STACKSIZE+1);
          IF ERR NEQ 0 THEN BEGIN STACKBASE:=0;
          ERRORMESS(ERR,0,0); GO TO PROCESSEXIT;END;
          SP[LOC]:=2;
          L:=L+1;
          M:=GETSPACE(STATEVECTORSIZE+1);
          SP[LOC]:=M&1[CRF]&DDPNVW[CDID];
          SP[MOC]:=STATEVECTORSIZE;
          M:=M+1; SP[MOC]:=0; % THE STATE VECTOR IS INITIALIZED NOW
          FUNCLOC:=M;
          N:=0;
          L:=L+1;                   COMMENT READY FOR A PROG MKS;
          END ELSE % THERE IS ALREADY A STACK...USE IT
          BEGIN L:=STACKBASE;
          ST:=SP[LOC]+L;
          WHILE M:=AREG.DID NEQ IMKS AND M NEQ PROGMKS AND
               ERR=0 DO POP;%STRIP BACK TO LASTMARKSTACK
          IF M=IMKS THEN BEGIN N:=ST-STACKBASE;PUSH;
               END ELSE N:=AREG.BACKF;
          SP[LOC]:=ST-STACKBASE;L:=ST;
          END;
          CURLINE:=0;
          M:=GETSPACE(1); SP[MOC]:=T; %STORE PROG DESCRIPTOR
          SP[LOC]:=M&PROGMKS[CDID]&N[BACKPT]&1[CI];
          COMMENT JUST BUILT A PROGRAM MARKSTACK;
          GO TO EXECUTION;
          END;
     %------------SUB-CASE 1....CURRENTMODE=XEQMODE--------------
          COMMENT RECOVERY FROM A TIME-OUT;
          GO TO EXECUTION;
     %----------- SUB-CASE 2....CURRENTMODE=FUNCMODE--------------
          COMMENT SYNTAX CHECK ONLY;
          IF ANALYZE(TRUE)=0 THEN;
     %----------- END OF SUB CASES-------------------------------
     END;
```

```
%--------------- CASE 2,....MODE=ALLOC-------------------------
    COMMENT NOTHING TO DO;
    ;
%--------------- CASE 3,... MODE=WRITEBACK----------------------
    COMMENT HAVE TO WRITE BACK ALL THE CHANGED VARIABLES;
    IF SYMBASE NEQ 0 THEN
        WRITEBACK;

%--------------- CASE 4,... MODE=DEALLOC------------------------
    ;


%--------------- CASE 5 ,... MODE=INTERROGATE-------------------
    COMMENT PRINT OUT THE PROGRAM STATUS VECTOR HERE;
    IF L:=STACKBASE+1 NEQ 1 THEN
        BEGIN COMMENT GT1=1 FOR SIV...=0 FOR SI;
        U:=GT1;
        L:=SP[LOC].SPF+1;M:=SP[LOC].SPF+L;
        WHILE M GTR L DO
            BEGIN N:=SP[MOC].LOCFIELD;N:=SP[NOC].SPF-1;
            % N IS LOCATION OF THE FUNCTION NAME
            ACCUM[0]:=SP[NOC];
            FORMROW(2,6,ACCUM,1,7);
            IF BOOLEAN(SP[MOC].SUSPENDED) THEN FORMWD(0,"3 S     ")
                ELSE FORMWD(0,"3       ");
            IF BOOLEAN(U) THEN % PRINT LOCAL VARIABLE NAMES
                BEGIN
                N:=SP[MOC].SPF+2;T:=SP[NOC]-2;
                FOR N:=N+4 STEP 2 UNTIL T DO
                    BEGIN ACCUM[0]:=SP[NOC];
                    FORMROW(0,1,ACCUM,1,7);
                    END;
                END;
            TERPRINT; M:=M-1;
            END;
        END;
    END;% OF THE CASE STATEMENT
%-------------END OF CASES---------------------------------------
IF FALSE THEN EXECUTION:
    BEGIN COMMENT EXECUTION LOOP;
    INTEGER LOOP;
    INTEGER INPUTIMS;
    LABEL BREAKKEY;
    LABEL SKIPPOP,XEQEPS;
    BOOLEAN XIT, JUMP;
    REAL POLWORD;
    DEFINE RESULT=RESULTD#;
    LABEL EXECEXIT, EVALQ, EVALQQ;
%%%
    COMMENT THERE IS A PROGRAM DESCRIPTOR AT THE TOP OF STACK;
    ERR:=0;
    L:=STACKBASE; ST:=L+SP[LOC];
    L:=L+1;FUNCLOC:=SP[LOC].SPF+1;
    T:=AREG;
    IF CURRENTMODE=XEQMODE THEN %AREG IS INTERRUPT MARK STACK
        BEGIN LASTMKS:=STACKBASE+T.BACKF;
        OLDDATA:=T.SPF; INPUTIMS:=T.QUADIN; POP;
        COMMENT MAY BE CURRENTLY EXECUTING A FUNCTION;
        L:=STACKBASE+1; L:=SP[LOC].SPF+1;
        IF (M:=SP[LOC].SPF) NEQ 0 THEN
            BEGIN M:=M+L; L:=SP[MOC].LOCFIELD;
            CURLINE:=SP[LOC].CIF;

            END;
        END
            ELSE LASTMKS:=ST;%AREG IS PROGRAM MARK STACK
    CURRENTMODE:=XEQMODE;
    L:=LASTMKS; T:=SP[LOC]; % T IS PROGRAM MARK STACK
    CINDEX:=T.CIF;          % CONTROL INDEX IN POLISH
    IF L:=T.SPF =0 THEN %PHONEY PROG DESC FROM FUNCTION CALL
        N:=POLTOP:=POLLOC:=0 ELSE
        BEGIN
        N:=POLLOC:=SP[LOC].SPF;
        POLTOP:=SP[NOC]
        END;
    IF ERR = 0 THEN % POP WORKED
        IF INPUTIMS=2 THEN BEGIN JUMP:=TRUE; GO TO EVALQ END ELSE
        IF INPUTIMS=1 THEN BEGIN JUMP:=TRUE; GO TO EVALQQ; END ELSE
        DO  BEGIN COMMENT EXECUTE UNTIL DONE OR TIME-OUT;
        IF CINDEX LSS POLTOP THEN %MORE TO EXECUTE IN POLISH
            BEGIN COMMENT GET NEXT POLISH TO EXECUTE;
            M:=(CINDEX:=CINDEX+1)+POLLOC;
            POLWORD:=T:=SP[MOC];
```

```
          CASE T.TYPEFIELD OF
               BEGIN %-------TF=0 (REPLACEMENT)-------------
                    BEGIN %MAY BE A LOCAL OR A GLOBAL VARIABLE
                    DEFINE STARTSEGMENT=#; %//////////////////////
                    PUSH; IF ERR NEQ 0 THEN GO TO SKIPPOP;
                    N:=T.LOCFIELD;
                    IF BOOLEAN(T.OPTYPE) THEN %A LOCAL VARIABLE
                         BEGIN M:=FUNCLOC;%FIND LAST FMKS
                         M:=SP[MOC].SPF+M;
                         N:=SP[MOC].LOCFIELD+N; END;
                    U:=SP[NOC]; U.LOCFIELD:=N; AREG:=U;
                         IF U.DATADESC=0 THEN ERR:=NONCEERROR;
                         COMMENT PROBABLY MIXUP WITH FUNCTION NAMES
                         AND NAMES OF LOCAL SUSPENDED VARIABLES;
               END;
               %-------------FUNCTION CALL----------------
%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
BEGIN COMMENT SET UP STACK FOR A FUNCTION CALL;
REAL U,V,NARGS,D;
INTEGER I,FLOC;
LABEL TERMINATE;
COMMENT
     MONITOR PRINT(D,L,M,N,FLOC,SP,LASTMKS);%::::::::::::::::::::
   FLOC:=N:=T.LOCFIELD;
   IF BOOLEAN(SP[NOC].DATADESC) THEN BEGIN ERR:=NONCEERROR;
      GO TO TERMINATE;END;%SUSPENDED VAR CONFUSED WITH FUNCTION
   IF NOT BOOLEAN(SP[NOC].PRESENCE) THEN MAKEFUNCTIONPRESENT(N);
   D:=SP[NOC]; L:=LASTMKS;   %D IS THE DESC, L IS THE PROG MKS
   SP[LOC].CIF:=CINDEX; %SAVE CURRENT POLISH LOCATION
   L:=STACKBASE+1; L:=SP[LOC].SPF+1;
   M:=SP[LOC].SPF;
   IF N:=M+L NEQ L THEN %THERE IS A NESTED CALL
      IF NOT BOOLEAN(SP[NOC].SUSPENDED) THEN
      BEGIN N:=SP[NOC].LOCFIELD;SP[NOC].CIF:=CURLINE;END;


   SETFIELD(GTA,0,8,0); SETFIELD(GTA,8,8,0); %INITIALIZE GTA
   NARGS:=D.NUMBERARGS;
   FOR I:=1 STEP 1 UNTIL NARGS DO
      IF BOOLEAN((T:=AREG).DATADESC) THEN
           BEGIN
           IF BOOLEAN(T.NAMED) THEN %MAKE A COPY
           COMMENT YOU COULD MAKE A CALL BY NAME HERE;
                BEGIN U:=GETSPACE(V:=(NUMELEMENTS(T)+T.RF));
                SPCOPY(T.SPF,U,V); T.NAMED:=0; T.SPF:=U;
                T.BACKP:=0;
                END ELSE %NO NEED TO MAKE A COPY
           AREG.PRESENCE:=0;
           POP; GTA[I-1]:=T; %SAVE THE DESCRIPTOR FOR LATER USE
           END ELSE ERR:=SYSTEMERROR;
   IF (N:=M+1) GEQ STATEVECTORSIZE THEN ERR:=DEPTHERROR;
   IF ERR NEQ 0 THEN GO TO TERMINATE;
   SP[LOC].SPF:=N;
   PUSH;AREG:=OLDDATA&(LASTMKS-STACKBASE)[BACKPT]&IMKS[CDID];
   OLDDATA:=0; %REINITIALIZE OLDDATA CHAIN FOR THIS FUNCTION
   %NOW SET UP THE FUNCTION MARK STACK.

   M:=N+L;PUSH;SP[MOC]:=D.SPF&ST[CLOCF];
   M:=D.SPF; M:=M+2; % M IS LOC OF LOC OF FIRST LINE
   AREG:=0&FLOC[CSPF]&((LASTMKS:=ST)-STACKBASE-1)[BACKPT]&
      (U:=SP[MOC]-D.SPF)[CCIF]&FMKS[CDID]; % FUNCTION MKS
   CURLINE:=U;

   U:=(U-6)/2; % U IS THE NUMBER OF LOCALS, LABELS, AND ARGS
   M:=M+5; % M IS ON THE FIRST DESC OF THE FIRST LAB, LOC,...
   FOR I:=1 STEP 1 UNTIL U DO % GET DESCRIPTORS INTO THE STACK
      BEGIN IF SP[MOC] NEQ 0 THEN %MAKE UP THE DESC
           BEGIN L:=GETSPACE(1); SP[LOC]:=SP[MOC];
           T:=L&DDPUSW[CDID]&0[CCIF]
           END ELSE
           T:=NULLV;
      PUSH; M:=M+2;
      AREG:=T; %A SINGLE LOCAL
      END;
   %COPY OVER THE ARGUMENTS
   FOR I:=1 STEP 1 UNTIL NARGS DO %COPY OVER
      BEGIN M:=D.SPF; %M IS THE LOCATION OF THE LABEL TABLE.
      M:=M+2+I; %M IS LOCATION OF REL LOCATION OF VARIABLE
      M:=SP[MOC];
      N:=LASTMKS+M;
      SP[NOC]:=GTA[I-1]
      END;
   %PUT IN A PHONEY PROG DESC TO START THINGS OFF
```

```
      PUSH;  IF ERR NEQ 0 THEN GO TO TERMINATE;                                037534
      AREG:=0&4094[CCIF]&(LASTMKS-STACKBASE)[BACKPT];                          037534
      LASTMKS:=ST; POLTOP:=POLLOC:=0;                                          037534
      TERMINATE:                                                               037535
      END;                                                                     037535
 %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&                   037535
                           %-------END OF LOAD FUNCTION FOR CALL-----          037539
                    %--------------TF=2 (CONSTANT)--------------------         037540
                    BEGIN PUSH; IF ERR=0 THEN BEGIN                            037541
                         N:=POLWORD.LOCFIELD;AREG:=SP[NOC];END;                037541
                    END;                                                       037541
                    %--------------TF=3 (OPERATOR)-----------------            037550
                    COMMENT SEQUENCE NUMBERS CORRESPOND TO OPERATOR            037551
                    ASSIGNMENT NUMBER;                                         037552
                    BEGIN IF T.OPTYPE=MONADIC THEN                             037552
                         BEGIN PUSH;IF ERR=0 THEN AREG:=0; END;                037552
                         CASE T.LOCFIELD OF                                    037553
 BEGIN %--------------- OPERATE ON STACK--------------------                   037554
  COMMENT EACH EXECUTION PROCEDURE SETS RESULT TO THE                          037555
    DESCRIPTOR OF THE RESULT OF THE OPERATION.                                 037555
     AREG AND BREG ARE THE LEFT AND RIGHT-HAND OPERANDS AND                    037555
     ARE ACTUALLY THE TOP TWO DESCRIPTORS ON THE STACK.                        037555
     IF AREG IS ZERO, THE OPERATOR IS TAKEN TO BE MONADIC.;                    037555
 ;                                                                             038000
 ;                                                                             038010
 ;                                                                             038020
 ;                                                                             038030
      %-------------------- REPLACEMENT OPERATOR--------------                 038040
      BEGIN DEFINE STARTSEGMENT=#; %//////////////////////////////             038041
      IF NOT BOOLEAN(L:=AREG.NAMED) THEN % SHOULD BE LOCAL VARIABLE            038041
          AREG.NAMED:=1; % DONT LET IT BE FORGOTTEN.                           038041
                                                                               038041
      IF BOOLEAN((T:=AREG).PRESENCE) AND T.SPF NEQ 0 THEN                      038042
          OLDDATA:=CHAIN(T,OLDDATA);                                           038042
      M:=T.LOCFIELD;                                                           038043
                                                                               038043
      IF(RESULT:=BREG).SPF = 0 THEN U:=T:=0 ELSE                               038043
          U:=GETSPACE(T:=NUMELEMENTS(RESULT)+RESULT.RF);                       038044
      SPCOPY(RESULT.SPF,U,T);                                                  038045
      RESULT.SPF:=U; RESULT.NAMED:=L; %L IS 0 FOR LOCALS                       038045
      GT1:=IF BOOLEAN((U:=SP[MOC]).PRESENCE) THEN U.BACKP ELSE 0;              038045
      SP[MOC]:=RESULT&GT1[CLOCF];                                              038045
      IF BOOLEAN(L) AND GT1=0 THEN %CHECK FOR GLOBAL                           038046
          BEGIN M:=M-1;IF(SP[MOC].SUSPENDVAR=0)THEN SP[MOC].CHANGE:=1;         038046
                                                                               038046
          END;                                                                 038046
      RESULT.NAMED:=1; %KEEP "PUSH" FROM TOSSING THE DATA                      038046
      END;                                                                     038047
      %-------TRANSFER OPERATOR---------------------------                     038050
      BEGIN DEFINE STARTSEGMENT=#; %//////////////////////////////             038051
      SCRATCHAIN(OLDDATA);OLDDATA:=0;                                          038051
      IF BOOLEAN(T.OPTYPE) THEN ST:=ST-1; %GET RID OF PHONEY TOP               038052
      L:=FUNCLOC;                                                              038052
      IF SP[LOC] NEQ 0 THEN STEPLINE(TRUE) ELSE                                038053
          ERR:=SYNTAXERROR;                                                    038054
      GO TO SKIPPOP;                                                           038055
      END;                                                                     038056
      BEGIN %---------------COMPRESSION------------------------------          038060
      DEFINE STARTSEGMENT=#; %////////////////////////////////                 038060
      L:=ST-2; IF T.OPTYPE=MONADIC THEN COMPRESS(BREG,SP[LOC],AREG)            038060
          ELSE COMPRESS(AREG,SP[LOC],BREG); COMMENT A/B HAS BEEN               038060
          STACKED AS B,A,NULL...A/[I] B HAS BEEN STACKED AS B,I,A;             038060
      END;                                                                     038060
      ARITH(3);  %OPERATION IS DIVIDE                                          038070
      ;                                                                        038079
 ;                                                                             038090
 %--------------QUAD INPUT-----------------------------------                  038100
   EVALQ:  BEGIN LABEL EVALQUAD;                                               038100
      IF JUMP THEN BEGIN JUMP:=FALSE; GO TO EVALQUAD END;                      038100
      CURRENTMODE:=INPUTMODE;                                                  038100
      FORMWD(3,"3[]:    "); INDENT(0);                                         038100
                                                                               038100
      IMS(2); % SETUP MARKSTACK FOR QUAD EXIT                                  038100
      IF ERR NEQ 0 THEN GO TO SKIPPOP;                                         038100
      GO TO EXECEXIT; % EXIT TO MONITOR TILL INPUT IS COMPLETE                 038100
 EVALQUAD: %LOOK AT BUFFER TO SEE WHAT CAME IN                                 038101
      BEGIN                                                                    038101
      IF NOT SCAN THEN BEGIN CINDEX:=CINDEX-1;GO TO SKIPPOP;END;               038101
      IF NOT SETUPLINE THEN CINDEX:=CINDEX-1;%MAKE THEM REDO IT                038101
          GO TO SKIPPOP;                                                       038102
          END;                                                                 038102
      END;                                                                     038105
      BEGIN % -----EVALUATE SUBSCRIPTS--------------                           038110
```

15

```
    DEFINE STARTSEGMENT=#; %///////////////////////////////////
    T:=AREG; L:=BREG.SPF;
    IF BOOLEAN(T.SCALAR) THEN BEGIN ERR:=DOMAINERROR;GO TO SKIPPOP;END;
    U:=SP[LOC]; % GET # OF SUBSCRIPTS
    IF U GTR 32 THEN ERR:=INDEXERROR ELSE
    BEGIN
    IF U GTR 0 THEN BEGIN
    IF T.PRESENCE NEQ 1 THEN % GET ARRAY INTO SP
        BEGIN N:=T.LOCFIELD;
        IF (T:=SP[NOC]).PRESENCE NEQ 1 THEN
                BEGIN T:=GETARRAY(T); SP[NOC]:=T END;
        T.LOCFIELD:= N;
        END;
    IF ERR=0 THEN % NOW EVALUATE

        RESULT:=SUBSCRIPTS(L:=(IF T.LOCFIELD=0 THEN OUTOF
            ELSE INTO),T,U);
        IF L=INTO THEN BEGIN

            CINDEX:=CINDEX+1;END; % SKIP OVER REPLACE OP
    END ELSE % NO SUBSCRIPTS
        BEGIN BREG:=T; ST:= ST-1; GO TO SKIPPOP;
        END; % DON≤T LET THE DESC. IN T BE POPPED.
    U:=U+2; % # OF THINGS TO POP
    FOR N:=1 STEP 1 UNTIL U DO POP;
    IF L=OUTOF THEN PUSH; AREG:=RESULT;

    GO TO SKIPPOP;
    END;
    END;
;
;
%=============QQUAD INPUT=================================
  EVALQQ:  BEGIN LABEL EVALQQUAD;
    IF JUMP THEN BEGIN JUMP:=FALSE; GO TO EVALQQUAD END;
    CURRENTMODE:=INPUTMODE;
    IMS(1); % SETUP MARKSTACKS FOR QQUAD EXIT
    IF ERR NEQ 0 THEN GO TO SKIPPOP;
    GO TO EXECEXIT;
EVALQQUAD:  % BUFFER CONTAINS THE INPUT STRING
    IF (L:=LENGTH(BUFFER,TRUE))NEQ 0 THEN BEGIN %L IS # CHAR INPUT
    N:=ENTIER((L+7) DIV 8); % FIND NUMBER OF WORDS
    M:=GETSPACE(N+1); % GET SPACE FOR THE VECTOR IN SP
    TRANSFERSP(INTO,SP,M+1,BUFFER,0,N);
    SP[MOC]:=L; % STORE LENGTH OF VECTOR
    RESULT:=M&1[CRF]&DDPUVC[CDID]; % SET UP DESCRIPTOR
    END ELSE RESULT:=NULLV;% NOTHING WAS INPUT
    PUSH; IF ERR=0 THEN AREG:=RESULT;
    GO TO SKIPPOP;
    END;
    RESULTD := SEMICOL;  %CONVERSION CONCATENATION
    COMMAP;     %CATENATE
    BEGIN%---------INNER PRODUCT (PERIOD)-------------------
    M:=(CINDEX:=CINDEX+2) + POLLOC; T:=SP[MOC];M:=M-1;U:=SP[MOC];
    PERIOD(AREG,BREG,U.LOCFIELD,T.LOCFIELD);
    END;
    ARITH(4);   %*
;
;

    ARITH(17);  %AND
    ARITH(18);  %OR
    ARITH(9);  %NOT
    ARITH(11);  %LESS:THAN
    ARITH(16);  %LEQ
    ARITH(12);  %=
    ARITH(13);  %GEQ
    ARITH(14);  %GREATER-THAN
    ARITH(15);  %NEQ
    ARITH(8);   %MAX/CEIL
    ARITH(7);   %MIN/FLOOR
    ARITH(6);   %RESD/ABS
    IF T.OPTYPE=MONADIC THEN GO TO XEQEPS ELSE MEMBER; %MEMBERSHIP
    RHOP;       %RHO
    IOTAP;      %IOTA
;
   REDUCESORTSCAN(0,BREG,AREG,4);  %REVERSAL;
   BEGIN %----------EXPANSION----------------------
     DEFINE STARTSEGMENT=#; %//////////////////////////////
     L:=ST-2; IF T.OPTYPE=MONADIC THEN EXPAND(BREG,SP[LOC],AREG)
      ELSE EXPAND(AREG,SP[LOC],BREG); COMMENT A EXPN B HAS BEEN
     STACKED AS B,A,NULL WHILE A EXPN [I] B IS STACKED AS B,I,A;
   END;
    RESULTD:=BASEVALUE;  %BASE VALUE
    ARITH(10);   %COMB/FACT
  ;
    IF T.OPTYPE=MONADIC THEN ARITH(5) ELSE
           DYADICRNDM;  %RNDM
    IF T.OPTYPE=MONADIC THEN TRANSPOSE ELSE DYADICTRANS;%GUESS WHAT
    RESULTD := REPRESENT;  %REPRESENTATION
    ARITH(45);  %CIRCLE--TRIGONOMETRIC FUNCTIONS
  ;
  ;
    ARITH(0);    %ADD
    ARITH(2);    %SUBTRACT
    ARITH(1);    %MULTIPLY
    %-----------------DISPLAY------------------------------------------

    BEGIN DEFINE STARTSEGMENT=#; %///////////////////////////////////
    IF BREG.SPF=0 THEN FORMROW(3,0,ACCUM,2,0) ELSE %FOR A NULL
    IF BOOLEAN((RESULT:=BREG).DATADESC)THEN %THIS IS A DATA DESC
       IF BOOLEAN(RESULT.PRESENCE) AND M:=RESULT.SPF NEQ 0 THEN
          IF BOOLEAN(RESULT.SCALAR) THEN
              BEGIN NUMBERCON(SP[MOC],ACCUM);
              FORMROW(3,0,ACCUM,2,ACOUNT)
              END
          ELSE %A VECTOR
          IF L:=RESULT.RF NEQ 0 THEN % SOMETHING TO PRINT
             IF BOOLEAN(RESULT.CHRMODE) THEN DISPLAYCHARV(RESULT)
             ELSE
             BEGIN RESULT:=M:=GETSPACE(L+1);
             SP[MOC]:=L; RESULT.RF:=1; RESULT.DID:=DDPUVW;
             AREG:=RESULT;
             FOR T:=1 STEP 1 UNTIL L DO
                 BEGIN M:=M+1; SP[MOC]:=1
                 END;
             DISPLAY(AREG,BREG);
             RESULT:=BREG;
             END ELSE TERPRINT
          ELSE TERPRINT
       ELSE ; %PROBABLY A FUNCTION....DONT DO ANYTHING
   IF BREAKFLAG THEN %USER HIT BREAK DURING OUTPUT
      GO TO BREAKKEY;
   POP; GO TO SKIPPOP;
   END;
    BEGIN % ----------------REDUCTION--------------------------------------
    M:=(CINDEX:= CINDEX+1) + POLLOC; % FIND OPERATION IN POLISH
    IF (T:=SP[MOC]).TYPEFIELD NEQ 3 THEN ERR:=SYSTEMERROR
      ELSE REDUCESORTSCAN(T.LOCFIELD,BREG,AREG,1);
    END;
   BEGIN %--------ROTATION---------------------------
     DEFINE STARTSEGMENT=#; %////////////////////////////////////////
     L:=ST-2; IF T.OPTYPE=MONADIC THEN
        REDUCESORTSCAN(BREG,SP[LOC],AREG,5) ELSE
        REDUCESORTSCAN(AREG,SP[LOC],BREG,5); COMMENT A ROT B IS
     STACKED AS B,A,NULL WHILE A ROT [I] B IS STACKED AS B,I,A;
   END;
    ARITH(21);  %LOG
   REDUCESORTSCAN(0,BREG,AREG,2);  % SORTUP
   REDUCESORTSCAN(-1,BREG,AREG,2); %SORTDN
   BEGIN %------------SCAN-------LIKE REDUCTION--------------
    DEFINE STARTSEGMENT=#; %/////////////////////////////////////////
     M:=(CINDEX:=CINDEX+1) + POLLOC;  %FIND OPERATOR IN POLISH
     IF (T:=SP[MOC]).TYPEFIELD NEQ 3 THEN ERR:=SYSTEMERROR
        ELSE REDUCESORTSCAN(T.LOCFIELD,BREG,AREG,3);
     END;
    ARITH(19);   %NAND
    ARITH(20);    %NOR
    IF (T:=BREG).RF NEQ 0 THEN RESULT:=SUBSCRIPTS(2,T,T,RF)
        ELSE ERR:=RANKERROR; % OPERATION IS TAKE
    IF (T:=BREG).RF NEQ 0 THEN RESULT:=SUBSCRIPTS(3,T,T,RF)
        ELSE ERR:=RANKERROR; % OPERATION IS DROP
    %----------------------XEQ-----------------------------------
XEQEPS:   BEGIN DEFINE STARTSEGMENT=#; %///////////////////
    IF AREG NEQ 0 THEN ERR:=SYNTAXERROR %MUST BE MONADIC ONLY
    ELSE IF (T:=BREG).RF NEQ 1 OR     %MUST BE A VECTOR
    NOT BOOLEAN(T.CHRMODE) THEN ERR:=DOMAINERROR %MUST BE CHAR STRING
    ELSE IF U:=NUMELEMENTS(T) GTR MAXBUFFSIZE THEN ERR:=LENGTHERROR
    ELSE BEGIN
        M:=GT1; % # OF CHARACTERS SET BY NUMELEMENTS
        INITBUFF(BUFFER,MAXBUFFSIZE);RESCANLINE;
        TRANSFERSP(OUTOF,SP,T.SPF+1,BUFFER,0,U);
        IF(U:=U×8-M) GTR 0 THEN SETFIELD(BUFFER,M,U,"       ");
        IF T.SPF=0 OR NOT SCAN THEN RESULT:=0&1[CRF]&DDPUVW[CDID]% NULL
        ELSE BEGIN POP;IF SETUPLINE THEN; GO TO SKIPPOP;END
          END; END;
    END; %--------------END OF OPERATION ON STACK------------------
```

```
                        POP;POP;PUSH;IF ERR=0 THEN AREG:=RESULT;
SKIPPOP:                END OF TYPEFIELD EQUALS OPERATOR;
                    %-------TF=4 (LOCAL VARIABLE)----------
                    BEGIN COMMENT MOVE DESCRIPTOR UP TO TOP;
                    DEFINE STARTSEGMENT=#; %/////////////////
                    N:=T.LOCFIELD;M:=FUNCLOC;M:=SP[MOC]+M;

                    N:=SP[MOC].LOCFIELD+N;
                    T:=SP[NOC]; T.NAMED:=1; %KEEP FROM THROWING AWAY
                    PUSH; AREG:=T;
                    END;
                    %-------TF=5 (OPERAND)-------------------
                    BEGIN PUSH; IF ERR=0 THEN BEGIN
                        N:=POLWORD.LOCFIELD; U:=SP[NOC];
                         IF U.DATADESC=0 THEN ERR:=NONCEERROR ELSE
                        IF U.PRESENCE NEQ 1 THEN BEGIN
                        U:=GETARRAY(U); SP[NOC]:=U END;
                    U.LOCFIELD:=0;
                        AREG:=U; END;
                    END;
                END;  % OF CASE STMT TESTING TYPEFIELD
            END % OF TEST FOR CINDEX LEQ POLTOP
            ELSE % WE ARE AT THE END OF THE POLISH
            BEGIN COMMENT LASTMKS CONTAINS THE LOCATION
            OF THE LAST MARK STACK.  GET MARK STACK AND CONTINUE;

            SCRATCHAIN(OLDDATA); OLDDATA:=0;
            L:=LASTMKS;M:=(U:=SP[LOC]).BACKF+STACKBASE;T:=SP[MOC];
            IF T.DID=IMKS AND T.QUADIN=3 THEN %SINGLE LINE DONE
                IF(RESULT:=AREG)=T THEN ERR:=SYNTAXERROR%NO RESULT
                ELSE BEGIN RESULT.NAMED:=0;%MAKE NEW COPY
                IF BOOLEAN(RESULT.SCALAR) THEN
                    BEGIN M:=GETSPACE(2);L:=RESULT.SPF;
                    RESULT.SPF:=M+1;SP[MOC]:=RESULT;
                    M:=M+1;SP[MOC]:=SP[LOC];
                    END ELSE % MAKE COPY OF A VECTOR
                    BEGIN M:=GETSPACE(1+(N:=RESULT.RF+NUMELEMENTS(
                        RESULT)));
                    L:=RESULT.SPF;RESULT.SPF:=M+1;
                    SP[MOC]:=RESULT; SPCOPY(L,M+1,N);END;


                FORGETPROGRAM(U);

                DO POP UNTIL ST LSS LASTMKS;%CUT BACK STACK TO IMS
                OLDDATA:=T.SPF;L:=LASTMKS:=T.BACKF+STACKBASE;
                AREG:=RESULT; % STORE EXECUTION RESULT OVER IMS
                CINDEX:=SP[LOC].CIF; M:= SP[LOC].SPF;
                POLLOC:=M:=SP[MOC].SPF; POLTOP:=SP[MOC];
                END ELSE
                BEGIN L:=FUNCLOC;M:=SP[LOC].SPF+L;
                IF M NEQ L AND NOT BOOLEAN(SP[MOC].SUSPENDED)THEN
                BEGIN
                IF 0=(LOOP:=(LOOP+1) MOD 5) THEN
                WRITE(TWXOUT,1,JIGGLE[*])[BREAKKEY:BREAKKEY];
                %THAT WAS TO CHECK FOR BREAK TO INTERRUPT A PROG
                    STEPLINE(FALSE)
                END
                ELSE BEGIN XIT:=TRUE;CURRENTMODE:=CALCMODE;
                    WHILE POPPROGRAM(OLDDATA,LASTMKS) DO;
                    END;
                END;
            END; %COMPLETION OF ONE POLISH EVALUATION (1 CELL)
        IF ERR NEQ 0 THEN % PUT OUT ERROR MESSAGE
            BEGIN
            DEFINE STARTSEGMENT=#; %////////////////////////////////
            COMMENT
            MONITOR PRINT(ST,L,M,SP,GTA,T);%::::::::::::::::::::::
            XIT:=TRUE;CURRENTMODE:=ERRORMODE;

            L:=POLLOC+1;
            TRANSFERSP(OUTOF,SP,(L:=SP[LOC].SPF)+1,BUFFER,
            0,MIN(MAXBUFFSIZE,ENTIER((SP[LOC]+7)DIV 8)));
            L:=FUNCLOC;M:=SP[LOC].SPF+L;
            GT1:=1;N:=SP[MOC].LOCFIELD;%LOCATION OF FMKS
            WHILE LASTMKS GTR N AND BOOLEAN (GT1) DO GT1:=IF
                POPPROGRAM(OLDDATA,LASTMKS)THEN 1 ELSE 0;
            IF M NEQ L AND NOT BOOLEAN(SP[MOC].SUSPENDED)THEN%GET LINE#
                BEGIN SP[LOC].RF:=SP[LOC].RF+1;%UP SUSPENDED COUNT
                L:=SP[NOC].SPF-1;%LOCATION OF FUNCTION NAME
                SETFIELD(GTA,0,1,0);
                GTA[0]:=SP[LOC];
                FORMROW(3,0,GTA,1,7);
                L:=SP[MOC].SPF; %BASE OF LABEL TABLE
                L:=L+CURLINE;
                T:=SP[LOC];

                %ALSO PUT THE FUNCTION INTO SUSPENSION
                IMS(4);SP[MOC].SUSPENDED:=1;SUSPENSION:=1;
                PUSHINTOSYMTAB(SP[MOC]);
                END ELSE T:=0;
           ERRORMESS(ERR,POLWORD.SPF,T);
             END;
        END UNTIL XIT;
BREAKKEY:    BEGIN BREAKFLAG:=FALSE;
             XIT:=TRUE;CURRENTMODE:=CALCMODE;
             L:=FUNCLOC;M:=SP[LOC].SPF+L;
             IF M NEQ L AND  NOT BOOLEAN(SP[MOC].SUSPENDED) THEN
                BEGIN SP[MOC].SUSPENDED:=1;SUSPENSION:=1;
                PUSHINTOSYMTAB(SP[MOC]);SP[LOC].RF:=SP[LOC].RF+1;
                M:=SP[MOC].LOCFIELD;%LOCATION OF FMKS IN STACK
                WHILE LASTMKS GTR M DO IF POPPROGRAM(OLDDATA,LASTMKS)
                THEN; LASTMKS:=M;IMS(4);
                END
        IF FALSE THEN
             END;
EXECEXIT:
        IF STACKBASE NEQ 0 THEN BEGIN
        L:=STACKBASE; SP[LOC]:=ST-L; %UPDATE SIZE OF STACK

        END;
    END OF EXECUTION LOOP;
PROCESSEXIT:
        IF BOOLEAN(POLBUG) THEN % DUMP SP
        IF MODE=XEQUTE OR MODE=3 OR MODE=6 THEN GO TO DEBUGSP;
    IF FALSE THEN
        BEGIN CASE 0 OF BEGIN
        EXPOVRL: SPOUT(3951200);
        INTOVRL: SPOUT(3951300);
        INDEXL:  SPOUT(3951400);
        FLAGL:   SPOUT(3951500);
        ZEROL:   SPOUT(3951600);
                      END;
        REALLYERROR:=1;
    DEBUGSP:
       WRITE(PRINT,MIN(15,PSRSIZE),PSR);
             BEGIN
             STREAM PROCEDURE FORM(A,B,N); VALUE N;
             BEGIN
             DI:=B; 15(DS:=8LIT" ");
             SI:=LOC N; DI:=B; DS:=8DEC; DI:=DI+3;
             SI:=A; 10(DS:=8CHR; DI:=DI+1);
             END;
             M:=MIN((NROWS+1)×SPRSIZE-1,MAXMEMACCESSES);
                FOR N:=0 STEP 10 UNTIL M DO
                BEGIN TRANSFERSP(OUTOF,SP,N,ACCUM,0,MIN(M-N,10));
                FORM(ACCUM,BUFFER,N);
                WRITE(PRINT,15,BUFFER[*]);
                END;
             END;
             IF POLBUG=0 OR BOOLEAN(REALLYERROR) THEN
        BEGIN
        ERRORMESS(IF ERR NEQ SPERROR THEN SYSTEMERROR ELSE ERR,0,0);
        SUSPENSION:=0;
        CURRENTMODE:=CALCMODE;
        REALLYERROR:=ERR:=0;
        END;
        END;
    END OF PROCESS PROCEDURE;
PROCEDURE ERRORMESS(N,ADDR,R); VALUE N,ADDR,R; REAL R;
    INTEGER N; REAL ADDR;
    BEGIN
    INTEGER STREAM PROCEDURE FORM(A,B); VALUE A;
        BEGIN LOCAL T,U;
        LABEL L,M;
        SI:=A;
        L: IF SC=" " THEN
             BEGIN SI:=SI+1; GO TO L;
             END;
        DI:=LOC T; DS:=2RESET; DS:=2SET;
        DI:=B; MESSIZE(U:=DI; DI:=LOC T; IF SC=DC THEN JUMP OUT TO M;
        SI:=SI-1; DI:=U; DS:=CHR; TALLY:=TALLY+1); M:
        FORM:=TALLY;
        END;
   ARRAY ERMES[0:13],B[0:MESSIZE/8];
  FILL ERMES[*] WITH
```

```
        "1         ",
        "5DEPTH   ",
        "6DOMAIN  ",
        "7EDITING",
        "5INDEX   ",
        "5LABEL   ",
        "6LENGTH ",
        "5NONCE   ",
        "4RANK    ",
        "6SYNTAX ",
        "6SYSTEM ",
        "5VALUE   ",
      "7SP FULL",
      "7FLYKITE";
    IF R NEQ 0 THEN
         BEGIN INDENT(R);CHRCOUNT:=CHRCOUNT-1
        END;
        FORMROW((IF R=0 THEN 2 ELSE 0),0,ERMES,N×8+1,
        ERMES[N].[1:5]);
        FORMWD(0,"6 ERROR");
         IF ADDR.[33:15] GEQ 512 THEN
         BEGIN
         FORMWD(0,"4 AT    ");
         FORMROW(1,1,B,0,FORM(ADDR,B))
         END;
         FORMWD(3,"1          ");
   END;
PROCEDURE LOADWORKSPACE(JOBNUM,NAME,IDENT); VALUE JOBNUM,NAME;
     REAL JOBNUM,NAME; ARRAY IDENT[0]; FORWARD;
PROCEDURE LOGINAPLUSER;
     BEGIN
     COMMENT LOG:IN THE CURRENT USER;
     COMMENT INPUT LINE IS IS THE BUFFER;
     LABEL EXEC, GUESS;
     DEFINE T=GT1#, J=GT2#,I=GT3#;
     PROCEDURE INITIALIZEPSR;
         BEGIN FOR I:=0 STEP 1 UNTIL PSRSIZE-1 DO
             PSRM[I] := 0;
         SEED:=STREAMBASE; ORIGIN:=1;
         FUZZ:=1@-11;
         LINESIZE:=72; DIGITS:=9;
         END;
     LADDRESS := ADDRESS := ABSOLUTEADDRESS;
     WORKSPACE:=WORKSPACEUNIT;
     ITEMCOUNT := EOB := 0;
     IF NEXTUNIT=WORKSPACEUNIT THEN % ESTABLISH A WORKSPACE
         BEGIN
                 WORKSPACE:=NEXTUNIT;
                 SEQUENTIAL(WORKSPACE);
                 INITIALIZEPSR;
                 I:=STORESEQ(WORKSPACE,PSR,PSRSIZE×8);
                 INITBUFF(OLDBUFFER,BUFFSIZE);

         END ELSE % WORKSPACE ASSIGNED
             I:=CONTENTS(WORKSPACE,0,PSR);
             FILL ACCUM[*] WITH "LOGGED I","N         ";
             FORMROW(0,1,ACCUM,0,9);
             I:=DAYTIME(ACCUM);
             FORMROW(1,1,ACCUM,0,I);
             SYMBASE:=STACKBASE:=0;
             CSTATION.APLOGGED:=1;
             CASE CURRENTMODE OF
                 BEGIN %-------CALCMODE-------------
                 ;COMMENT NOTHING TO DO ANYMORE;
                 %-------------XEQUTEMODE-----------
EXEC:
                  BEGIN FILL ACCUM[*] WITH "LAST RUN"," STOPPED";
                  FORMROW(3,0,ACCUM,0,16);
                 CURRENTMODE:=CALCMODE;
                  END;
                 %------------FUNCMODE---------------
                  BEGIN FILL ACCUM[*]WITH "CONTINUE"," DEFINIT",
                  "ION OF   ";
                  FORMROW(2,0,ACCUM,0,23); FORMROW(1,0,PSR,
                  FSTART×8,7);
                 CURLINE:=GT3:=TOPLINE(GT1:=FUNCPOINTER);
                 CHECKSEQ(CURLINE,GT3,INCREMENT); %GET INCREMENT
                 CURLINE:=CURLINE+INCREMENT; INDENT(-CURLINE);
                 FUNCSIZE:=SIZE(GT1);
                 END;
                 %------------INPUTMODE--------------ERRORMODE----
                  GO TO EXEC; GO TO EXEC;
                 END;
    GUESS:  %SHOULD BE BETTER PLACE BUT HERE IS WHERE OTHERS COME OUT
    STOREPSR;
    IF CURRENTMODE NEQ FUNCMODE THEN
    INDENT(0); TERPRINT;
    VARSIZE:=IF VARIABLES=0 THEN 0 ELSE SIZE(VARIABLES);
    END;
ROCEDURE APLMONITOR;
    BEGIN
    REAL T;
    INTEGER I;
    BOOLEAN WORK;
    LABEL AROUND, NEWUSER;
    LABEL CALCULATE,EXECUTEIT,FUNCTIONSTART,BACKAGAIN;
    LABEL CALCULATEDIT;
    I := CUSER := 1;
    T := STATION;
              BEGIN FILL ACCUM[*] WITH "ApL/B550","O UW COM"
              ,"PUTER SC","IENCE # ",VERSIONDATE;
              WORK:=TRUE;
              FORMROW(3,MARGINSIZE,ACCUM,0,40);
              INDENT(0); TERPRINT; CSTATION.APLHEADING:=1
              ; LOGINAPLUSER;
              END;
AROUND:

        BEGIN
        IF MAINTENANCE THEN;
        CASE CURRENTMODE OF
        BEGIN %-------CALCMODE-----------------------------------
        COMMENT HE MUST BE READ READY FOR THE CALCMODE STUFF;

            GO CALCULATE;
        %--------XEQUTE MODE---------------------------------
            GO TO EXECUTEIT;
        %---------FUNCMODE-----------------------------------
            GO TO FUNCTIONSTART;
        %----------INPUTMODE----------------------------------
        COMMENT REQUIRES INPUT;

             BEGIN COMMENT GET THE LINE AND GO BACK;
             STARTSCAN;
             CURRENTMODE:=XEQMODE;
            GO TO EXECUTEIT;
             END;
        %----------ERRORMODE---------------------------------
        GO TO BACKAGAIN;

        END; %OF CASES
        END;
    COMMENT GET HERE IF NOTHING TO DO;

    GO TO AROUND;
    CALCULATE:
        STARTSCAN;
ALCULATEDIT:
        ERR:=0; %AND DON"T RESET IT IN SCAN OR IN ANALYZE
        IF SCAN THEN
            IF RGTPAREN THEN MESSAGEHANDLER ELSE
        IF DELV THEN FUNCTIONHANDLER ELSE
            BEGIN COMMENT PROCESS CALCULATOR MODE REQUEST;
            MOVE(BUFFER,BUFFSIZE,OLDBUFFER);
            IF NOT BOOLEAN(SUSPENSION) THEN BEGIN %INITIALIZE USER
"%%
 %%

            SYMBASE:=STACKBASE:=0;
            END;
            PROCESS(XEQUTE);
            IF CURRENTMODE=CALCMODE THEN
ACKAGAIN:       BEGIN INDENT(0); TERPRINT;
                IF NOT BOOLEAN(SUSPENSION) THEN
                    BEGIN IF CURRENTMODE NEQ ERRORMODE THEN
                    PROCESS(WRITEBACK);
                    SP[0,0]:=0;NROWS:=-1;

%%
                    END;
                CURRENTMODE:=CALCMODE;
                END;
            END;
        IF EDITOG=1 THEN
                BEGIN  MOVE(OLDBUFFER,BUFFSIZE,BUFFER);
                RESCANLINE; EDITOG:=0; GO TO CALCULATEDIT;
                END;
        I:=0;
        GO AROUND;
    EXECUTEIT:
        PROCESS(XEQUTE); %GO BACK TO PROCESS FOR AWHILE
        IF CURRENTMODE=CALCMODE THEN GO TO BACKAGAIN;
        I:=0;
        GO AROUND;
    FUNCTIONSTART:
    IF SPECMODE = 0 THEN
        BEGIN %SEE IF A SPECIAL FUNCTION.
        STARTSCAN;
        IF SCAN AND RGTPAREN THEN MESSAGEHANDLER ELSE
        FUNCTIONHANDLER
        END ELSE
    FUNCTIONHANDLER;
        I:=0;
        GO AROUND
    END;
INTEGER PROCEDURE LENGTH(A,M);VALUE M; BOOLEAN M; ARRAY A[0];
    BEGIN
INTEGER STREAM PROCEDURE LENGT(A,M,L); VALUE M,L;
    BEGIN LOCAL T;
    LOCAL C,CC,TSI;        LABEL LAB;
    LOCAL AR; LABEL LAB2;
    SI:=LOC M; SI:=SI+7;
    IF SC="1" THEN
        BEGIN COMMENT LOOK FOR LEFT ARROW.;
        DI:=LOC AR; DS:=RESET; DS:=5SET;
        SI:=LOC L; DI:=LOC T; DI:=DI+1; DS:=7CHR;
        SI:=A;
        T(2(32(DI:=LOC AR; IF SC=DC THEN JUMP OUT 3 TO LAB;
        TALLY:=TALLY+1;
        C:=TALLY; TSI:=SI; SI:=LOC C;
        SI:=SI+7; IF SC="0" THEN
            BEGIN TALLY:=CC; TALLY:=TALLY+1; CC:=TALLY;
            TALLY:=0;
            END; SI:=TSI)));
        L(DI:=LOC AR; IF SC=DC THEN JUMP OUT TO LAB;
        TALLY:=TALLY+1; C:=TALLY; TSI:=SI; SI:=LOC C; SI:=SI+7;
        IF SC="0" THEN
            BEGIN TALLY:=CC; TALLY:=TALLY+1; CC:=TALLY; TALLY:=0
            END; SI:=TSI);
        LAB: SI:=LOC CC; DI:=LOC LENGT; DI:=DI+6; SI:=SI+7;
        DS:=CHR; SI:=LOC C; SI:=SI+7; DS:=CHR;
        END ELSE
        BEGIN
        SI:=LOC L; DI:=LOC T; DI:=DI+1; DS:=7CHR;
        SI:=A; T(2(SI:=SI+32)); SI:=SI+L;
        T(2(32(SI:=SI-1; IF SC NEQ " " THEN JUMP OUT 3 TO LAB2;
        TALLY:=TALLY+1; C:=TALLY; TSI:=SI; SI:=LOC C; SI:=SI+7;
        IF SC="0" THEN
            BEGIN TALLY:=CC; TALLY:=TALLY+1; CC:=TALLY; TALLY:=0
            END; SI:=TSI)));
        L(SI:=SI-1; IF SC NEQ" " THEN JUMP OUT TO LAB2;
        TALLY:=TALLY+1; C:=TALLY; TSI:=SI; SI:=LOC C; SI:=SI+7;
        IF SC="0" THEN
            BEGIN TALLY:=CC; TALLY:=TALLY+1; CC:=TALLY; TALLY:=0
            END; SI:=TSI);
        LAB2: GO TO LAB
        END
    END;
NTEGER I;
:=LENGT(A,M,BUFFSIZE×8);
.ENGTH:=IF M THEN I ELSE BUFFSIZE×8-I
   END;
OOLEAN PROCEDURE LABELSCAN(L,K); VALUE K; INTEGER K; ARRAY L[0];
    BEGIN REAL T;
    T:=ADDRESS;
    IF SCAN AND IDENT THEN
        BEGIN SETFIELD(ACCUM,1,1,0); TRANSFER(ACCUM,1,L,K×8,8);
        IF NOT(LABELSCAN:=(SCAN AND COLON)) THEN
            BEGIN ADDRESS:=T; EOB:=0; IF SCAN THEN;
            END;
        END
    END;
TREAM PROCEDURE MOVEWDS(A,N,B); VALUE N;
    BEGIN SI:=A; DI:=B; DS:=N WDS END;
NTEGER PROCEDURE DAYTIME(B); ARRAY B[0];
    BEGIN

    INTEGER D,H,M,MIN,Q,P,Y,TIME1;
    LABEL OWT;
    STREAM PROCEDURE FORM(A,DAY,MO,DA,YR,HR,MIN,AP);
        VALUE DAY,MO,DA,YR,HR,MIN,AP;
        BEGIN DI:=A;
        SI:=LOC DAY; SI:=SI+7;
        IF SC="0" THEN DS:=3LIT"SUN" ELSE
        IF SC="1" THEN DS:=3LIT"MON" ELSE
        IF SC="2" THEN DS:=4LIT"TUES" ELSE
        IF SC="3" THEN DS:=6LIT"WEDNES" ELSE
        IF SC="4" THEN DS:=5LIT"THURS" ELSE
        IF SC="5" THEN DS:=3LIT"FRI" ELSE DS:=5LIT"SATUR";
        DS:=4LIT"DAY "; SI:=LOC MO; DS:=2DEC;
        DS:=LIT"-"; SI:=LOC DA; DS:=2DEC; DS:=LIT"-";
        SI:=LOC YR; DS:=2DEC; DS:=2LIT" ";
        SI:=LOC HR; DS:=2DEC; DS:=LIT":"; SI:=LOC MIN;
        SI:=SI+6; DS:=2CHR; SI:=LOC AP; SI:=SI+7; DS:=LIT" ";
        DS:=CHR; DS:=LIT"M"
        END;
    TIME1:=TIME(1);
    Y:=TIME(0);
        D:=Y.[30:6]×100+Y.[36:6]×10+Y.[42:6];
        Y:=Y.[18:6]×10+Y.[24:6];
    FOR H:=31,IF Y MOD 4=0 THEN 29 ELSE 28,31,30,
       31,30,31,31,30,31,30 DO
        IF D LEQ H THEN GO OWT ELSE
             BEGIN D:=D-H; M:=M+1
             END;
    OWT:
    H:=TIME1 DIV 216000;
    MIN:=(TIME1 DIV 3600) MOD 60;
    IF M LSS 2 THEN
        BEGIN Q:=M+11; P:=Y-1
        END ELSE
        BEGIN Q:=M-1; P:=Y
        END;
    M:=M+1;
    FORM(B,TIME1:=((Q×26-2)DIV 10+D+P+P.[36:10]+1)MOD 7,
        M,D,Y,Q:= H MOD 12, Q:=MIN MOD 10+(MIN DIV 10)×64,
        IF H GEQ 12 THEN "P" ELSE 17);
    DAYTIME:=(IF TIME1=6 THEN 5 ELSE
        IF TIME1=5 THEN 3 ELSE
             IF TIME1=2 THEN 4 ELSE 3)+22;


    END;
PROCEDURE LOADWORKSPACE(NAME1,NAME2,IDENT); VALUE NAME1,NAME2;
    REAL NAME1,NAME2; ARRAY IDENT[0];
    BEGIN
    FILE DISK DISK(2,WDSPERREC,WDSPERBLK);
    INTEGER PROCEDURE RD(D,N,A);
        VALUE N; INTEGER N; FILE D; ARRAY A[0];
        BEGIN READ(D[N],WDSPERREC,A[*]);
        RD:=N+1;
        END;
    PROCEDURE LOADITEM(RD,D,ITEM);
        INTEGER PROCEDURE RD; FILE D;
        ARRAY ITEM[0];
        BEGIN
        DEFINE T=ITEM#;
    PROCEDURE GETALINE(C,S,L,B,RD,D,LEN);
        VALUE LEN; INTEGER C,S,L,LEN;
        ARRAY B[0]; INTEGER PROCEDURE RD; FILE D;
        BEGIN % GET 2 CHRS GIVING ENSUING CHAR COUNT
        INTEGER P;
        IF C GTR LEN-2 THEN
            IF C GTR LEN-1 THEN % READ A NEW RECORD AND TAKE 2 CHRS
                BEGIN
                S:=RD(D,S,B);
                C:=2;
               TRANSFER(B,0,L,6,2);
               END
            ELSE % 1 CHR LEFT ON LINE
                BEGIN
                TRANSFER(B,C,L,6,1);
                S:=RD(D,S,B);
                TRANSFER(B,0,L,7,1);
                C:=1;
                END
        ELSE % AT LEAST 2 CHARS REMAINING ON LINE
            BEGIN
            TRANSFER(B,C,L,6,2);
            C:=C+2;
            END;
        P:=0;
        IF L NEQ 0 THEN % SOMETHING LEFT IN FUNCTION
            BEGIN
    WHILE P LSS L DO
    IF (L-P) GTR (LEN-C) THEN % # OF CHARS IN LINE
       % EXTENDS INTO NEXT RECORD
        BEGIN
        TRANSFER(B,C,BUFFER,P,LEN-C); % FINISH OUT RECORD
        S:=RD(D,S,B);
        P:=P+(LEN-C); % AMOUNT READ SO FAR
        C:=0;
        END
    ELSE % ALL ON ONE RECORD
        BEGIN
        TRANSFER(B,C,BUFFER,P,L-P);
        C:=C+L-P;
        P:=L; % FINISHED
        END;
    END;
END OF GETALINE;
INTEGER S,K,L,M,C,LEN,SQ,PT,G,I,SIZE;
INTEGER HOLD;
LABEL SCALARL;
ARRAY U[0:1],B[0:WDSPERREC-1];
BOOLEAN TOG;
TRANSFER(T,0,U,0,7);
G:=GETFIELD(T,7,1);
IF VARSIZE GTR 0 THEN
    IF K:=SEARCHORD(VARIABLES,U,HOLD,7)=0 THEN
        IF K:=GETFIELD(U,7,1)=FUNCTION THEN TOG:=TRUE
        ELSE % NOT A FUNCTION IN THE SYMBOL TABLE
        IF G=FUNCTION THEN
            BEGIN
            DELETE1(VARIABLES,HOLD);
            IF K≠ARRAYDATA THEN RELEASEARRAY(U[1]);
            END
        ELSE TOG:=TRUE % DON-T CHANGE
    ELSE % NOT IN VARIABLES
        BEGIN
        VARSIZE:=VARSIZE+1;
        HOLD:=HOLD+K-1;
        END
ELSE VARSIZE:=1;
LEN:=(WDSPERREC-1)×8;
IF NOT TOG THEN % OK TO PUT INTO VARIABLES
    IF G=FUNCTION THEN % READ A FUNCTION  INTO VARIABLES
        BEGIN
        TRANSFER(T,0,U,0,9); % U HOLDS FUNCTION NAME,
        %NUMBER OF ARGUMENTS, AND WHETHER FN RETURNS A VALUE
        S:=T[1].LIBF1; % RECORD NUMBER
        M:=T[1].LIBF2; % WORD WITHIN RECORD
        SIZE:=T[1].LIBF3; % SIZE OF POINTERS TABLE
        PT:=NEXTUNIT;
        S:=RD(D,S,B);
        FOR I:=0 STEP 1 UNTIL SIZE-1 DO
            BEGIN
            TRANSFER(B,M×8,T,0,16);
            M:=M+2;
            IF M GEQ WDSPERREC-1 THEN
                BEGIN
                S:=RD(D,S,B);
                IF M GEQ WDSPERREC THEN
                    BEGIN
                    TRANSFER(B,0,T,8,8);
                         M:=1;
                    END
                ELSE M:=0;
                 END;
            STOREORD(PT,T,I);
            END; % HAVE FINISHED FILLIN G POINTERS TABLE
        IF VARIABLES=0 THEN BEGIN
            VARIABLES:=NEXTUNIT; TOG:=TRUE; %KEEP THE UNIT OPEN
            STOREORD(VARIABLES,U,HOLD); END;
        SEQUENTIAL (SQ:=NEXTUNIT);
        SETFIELD(U,FPTF,FFL,PT);
        SETFIELD(U,FSQF,FFL,SQ);
        STOREORD(VARIABLES.,U,HOLD);
        IF TOG THEN DELETE1(VARIABLES,HOLD+1);%REMOVE 1 EXTRA
        COMMENT NOW FILL IN SEQ STORAGE;
        IF M NEQ 0 THEN BEGIN
        M:=C:=0;
        S:=RD(D,S,B); % TEXT STARTS AT BEG. OF NEW RECORD
                         END;
        L:=1;

        WHILE L NEQ 0 DO
```

```
                    BEGIN
                    GETALINE(C,S,L,B,RD,D,LEN);
                    GT1:=STORESEQ(SQ,BUFFER,L);
                    END
                END
            ELSE
            IF G=ARRAYDATA THEN
                    IF T[1].INPTR=0 THEN % NULL VECTOR
                        GO SCALARL
                    ELSE
                BEGIN
                ARRAY DIMVECT[0:MAXBUFFSIZE];
                S:=T[1].INPTR; % RECORD NUMBER
                M:=T[1].DIMPTR; % LOC WITHIN RECORD
                C:=M×8;
                SIZE:=T[1].RF; % RANK
                S:=RD(D,S,B);
                GETALINE(C,S,L,B,RD,D,LEN);
                T[1].DIMPTR:=STORESEQ(WS,BUFFER,L);
                    % PUTS DIMVECT INTO WORKSPACE
                GETALINE(C,S,L,B,RD,D,LEN); % # BLOCKS
                SIZE:=L-1;
                FOR K:=0 STEP 2 UNTIL SIZE DO
                    BEGIN
                    GETALINE(C,S,L,B,RD,D,LEN);
                    SETFIELD(DIMVECT,K,2,STORESEQ(WS,BUFFER,L));
                    END; COMMENT THIS STORES THE VALUES OF THE
                        ARRAY INTO THE WORKSPACE, AND ALSO RECORDS
                        THE LOCATION WITHIN WS IN DIMVECT,TO BE STORED;
                T[1].INPTR:=STORESEQ(WS,DIMVECT,SIZE+1);
                IF VARIABLES=0 THEN VARIABLES:=NEXTUNIT;
                STOREORD(VARIABLES,T,HOLD);
                END
            ELSE % MUST BE A SCALAR
                    SCALARL:
                BEGIN
                 IF VARIABLES=0 THEN VARIABLES:=NEXTUNIT;
                STOREORD(VARIABLES,T,HOLD);
                END
        ELSE % WILL NOT REPLACE IN SYMBOL TABLE
            BEGIN
            FILL BUFFER[*] WITH " ","NOT REPL","ACED     ";
                TRANSFER(T,0,BUFFER,0,7);
                FORMROW(3,0,BUFFER,0,20);
            END;
        END LOADITEM;
    BOOLEAN STREAM PROCEDURE EQUAL(A,B);
        BEGIN SI:=A; DI:=B; SI:=SI+2; IF 7SC=DC THEN TALLY:=1;
        EQUAL:=TALLY
        END;
    INTEGER I,J,L,NDIR,N;
    LABEL MOVEVAR,SKIP;
    ARRAY T,U[0:1],D[0:WDSPERREC-1];
    FILL DISK WITH NAME1,NAME2; L:=RD(DISK,L,D);
    IF D[1] NEQ JOBNUM AND D[1] NEQ 0 THEN GO SKIP; % FILE LOCKED
    FOR I:=2 STEP 1 UNTIL 9 DO IF GETFIELD(D[I],1,7) NEQ C THEN GO SKIP;
    IF NDIR:=D[0] NEQ 0 THEN
    BEGIN N:=LIBSPACES+ENTIER(NDIR×2/(J:=WDSPERREC-1));
        IF(NDIR×2) MOD J NEQ 0 THEN N:=N+1;
        FOR I:=1 STEP 1 UNTIL NDIR DO
            BEGIN COMMENT GET FUNCTION OR VARIABLE NAME FROM LIB;
            IF WDSPERREC-J LSS 3 THEN
                IF WDSPERREC-J=1 THEN
                BEGIN L:=RD(DISK,L,D); J:=0; GO MOVEVAR
                END ELSE
                BEGIN TRANSFER(D,J×8,T,0,8); L:=RD(DISK,L,D);
                TRANSFER(D,0,T,8,8); J:=1
                END ELSE MOVEVAR:
                BEGIN TRANSFER(D,J×8,T,0,16); J:=J+2
                END;
            IF(IF IDENT[0]=0 THEN TRUE ELSE EQUAL(IDENT,T)) THEN
                BEGIN IF IDENT[0] NEQ 0 THEN I:=NDIR+1;
                LOADITEM(RD,DISK,T);
                END
            END;
    STOREPSR; % UPDATE SINCE HAVE ADDED TO VARIABLES
    END;
    IF FALSE THEN SKIP: FORMWD(1,"6BADFIL");
    EOB:=1;
    END OF LIBRARY LOAD;
PROCEDURE PURGEWORKSPACE(WS); VALUE WS; INTEGER WS;
    IF WORKSPACE NEQ 0 THEN
    BEGIN
```

```
        INTEGER I,J,K,V,L,G;                                                      08015020
        ARRAY T[0:1];                                                             08015030
        J:=SIZE(V:=VARIABLES)-1;                                                  08015040
        FOR I:=0 STEP 1 UNTIL J DO                                                08015050
            BEGIN K:=CONTENTS(V,I,T);                                             08015060
            IF GETFIELD(T,7,1)=FUNCTION THEN                                      08015070
            FOR L:=FPTF,FSQF DO    % GET RID OF STORAGE                           08015080
            IF G:=GETFIELD(T,L,FFL) NEQ 0 THEN RELEASEUNIT(G);                    08015090
            END;                                                                  08015100
        RELEASEUNIT(V);                                                           08015110
        VARIABLES:=0; VARSIZE:=0;                                                 08015120
        CURRENTMODE:=0; J:=SIZE(WS)-1;                                            08015122
        FOR I:=1 STEP 1 UNTIL J DO DELETE1(WS,I);                                 08015124
        STOREPSR;                                                                 08015130
        END;                                                                      08015140
    PROCEDURE ELIMWORKSPACE(WS); VALUE WS; INTEGER WS;                            08015150
        BEGIN LABEL QQQ; QQQ:                                                     08015152
        IF WORKSPACE NEQ 0 THEN                                                   08015155
            BEGIN                                                                 08015205
            PURGEWORKSPACE(WS); RELEASEUNIT(WS);                                  08015210
%                                                                                 08015220
        END ELSE SPOUT(8015222);                                                  08015222
        END;                                                                      08015225
    PROCEDURE SAVEWORKSPACE(NAME1,NAME2,LOCKFILE);                                08015300
        VALUE NAME1,NAME2,LOCKFILE;                                               08015305
        REAL NAME1,NAME2,LOCKFILE;                                                08015310
        BEGIN                                                                     08015320
        SAVE FILE DISK DISK [NAREAS:SIZEAREAS]                                    08015330
          (2,WDSPERREC,WDSPERBLK,SAVE 100);                                       08015340
        INTEGER PROCEDURE WR(D,N,A); VALUE N; INTEGER N;                          08015350
            FILE D; ARRAY A[0];                                                   08015360
            BEGIN REAL STREAM PROCEDURE CON(A); VALUE A;                          08015370
                BEGIN SI:=LOC A; DI:=LOC CON; DS:=8DEC END;                       08015380
            STREAM PROCEDURE CLEANER(A);                                          08015382
                BEGIN DI:=A; WDSPERREC(DS:=8LIT","") END;                         08015384
            A[WDSPERREC-1]:=CON(N);                                               08015390
            WRITE(D[N],WDSPERREC,A[*]);                                           08015400
            WR:=N+1; CLEANER(A);                                                  08015410
            END;                                                                  08015420
        PROCEDURE PUTAWAY(C,J,WR,D,N,M,B,L); VALUE L,J;                           08015430
            INTEGER L,C,J,N,M;                                                    08015435
        ARRAY B[0]; INTEGER PROCEDURE WR; FILE D;                                 08015440
            BEGIN INTEGER P,K;                                                    08015450
            IF C+2 GTR L THEN                                                     08015460
                BEGIN TRANSFER(J,6,B,C,1); N:=WR(D,N,B); C:=1;                    08015470
                TRANSFER(J,7,B,0,1);                                              08015480
                END ELSE                                                          08015490
                BEGIN TRANSFER(J,6,B,C,2); C:=C+2                                 08015500
                END;                                                              08015510
            WHILE J NEQ 0 DO                                                      08015520
            IF J GTR K:=(L-C) THEN                                                08015530
                BEGIN TRANSFER(BUFFER,P,B,C,K);                                   08015540
                N:=WR(D,N,B); J:=J-K; C:=0; P:=P+K                                08015550
                 END ELSE                                                         08015560
                BEGIN TRANSFER(BUFFER,P,B,C,J); C:=C+J; J:=0                      08015570
                END;                                                              08015580
            IF C=L THEN                                                           08015590
                BEGIN N:=WR(D,N,B); C:=0                                          08015600
                END;                                                              08015606
            END;                                                                  08015609

        PROCEDURE MOVETWO(U,B,M,WR,L,D);                                          08015610
            ARRAY U,B[0]; INTEGER M,L; INTEGER PROCEDURE WR; FILE D;              08015612
            BEGIN                                                                 08015615
            COMMENT PUTS 2 WORDS OF U IN B AND WRITES ON D IF  A FULLRECORD;      08015618
            TRANSFER(U,0,B,M×8,16); % CONTENTS OF U INTO B                        08015621
            M:=M+2;                                                               08015624
            IF M GEQ WDSPERREC-1 THEN % FULL RECORD                               08015627
                BEGIN                                                             08015630
                L:=WR(D,L,B);                                                     08015633
                IF M GEQ WDSPERREC THEN % 1 OVER FULL RECORD                      08015636
                                                                                  08015639
                    BEGIN                                                         08015640
                    TRANSFER(U,8,B,0,8);                                          08015642
                    M:=1;                                                         08015645
                    END                                                           08015648
                ELSE M:=0;                                                        08015651
                END;                                                              08015654
            END OF MOVETWO;                                                       08015657
            INTEGER H,Q,M,N,I,L,S,J,K,LINE,MAX,PT,SQ,C,LEN,W;                     08015660
            REAL LSD,STP;                                                         08015663
            LABEL SKIP;                                                           08015666
            ARRAY T,U[0:1],DIR,B,SEX[0:WDSPERREC];                                08015669
                                                                                  08015672
```

```
N:=LIBSPACES+ENTIER((S:=SIZE(VARIABLES))×2/(WDSPERREC-1));
IF (S×2) MOD (WDSPERREC-1) NEQ 0 THEN N:=N+1; % ADJUST
LEN:=(WDSPERREC-1)×8;
FILL DISK WITH NAME1,NAME2;
DIR[0]:=S; % SIZE OF SYMBOL TABLE
IF BOOLEAN (LOCKFILE) THEN DIR[1]:=JOBNUM;
S:=S-1;
L:=WR(DISK,L,DIR); % FIRST LINE CONTAINS # OF ENTRIES IN
COMMENT SYMBOL TABLE AND LOCK INFORMATION;
FOR I:=0 STEP 1 UNTIL S DO
   BEGIN
    J:=CONTENTS(VARIABLES,I,T); % RETURNS VALUE OF I-TH LOC
    % IN VARIABLES INTO T
    IF GT2:=GETFIELD(T,7,1)=FUNCTION THEN
       BEGIN
       PT:=GETFIELD(T,FPTF,FFL); % FUNCTION POINTER FIELD
       SQ:=GETFIELD(T,FSQF,FFL); % FUNCTION SEQUENTIAL FIELD
       %PT=# OF ORDERED STORAGE UNIT CONTAINING HEADER&POINTE
       %SQ=# OF SEQ STORAGE UNIT CONTAINING TEXT
       MAX:=SIZE(PT);
       T[1].LIBF1:=N; % RECORD #
       T[1].LIBF2:=M; % LOC WITHIN RECORD
       T[1].LIBF3:=MAX; % SIZE OF  POINTERS TABLE;
       % SAVE ENOUGH ROOM FOR THE ENTIRE POINTERS TABLE
       H:=ENTIER(GT1:=(M+MAX×2)/(WDSPERREC-1));
       H:=IF GT1 NEQ H THEN H+N+1 ELSE H+N;
       U[0]:=0;
       J:=SEARCHORD(PT,U,LINE,8); % LOOK FOR ALL ZEROS
       IF J=2 THEN GO SKIP;
       FOR W:=0 STEP 1 UNTIL LINE-1 DO
           %MOVE LOCALS AND LABELS INTO THE SAVE FILE
           BEGIN
       J:=CONTENTS(PT,W,U);
           MOVETWO(U,B,M,WR,N,DISK);
           END;
           FOR LINE:=LINE STEP 1 UNTIL MAX-1 DO
              BEGIN

              J:=CONTENTS(PT,LINE,U);
                  GT1:=U[1];
              U[1]:=LINE-W;
              MOVETWO(U,B,M,WR,N,DISK); % POINTERS TABLE
                  J:=CONTENTS(SQ,GT1,BUFFER);
              PUTAWAY(C,J,WR,DISK,H,Q,SEX,LEN); % TEXT
              END;
           PUTAWAY(C,0,WR,DISK,H,Q,SEX,LEN);
     SKIP:
      Q:=C DIV 8;
           IF C MOD 8 NEQ 0 THEN Q:=Q+1;
           IF Q=WDSPERREC-1 THEN
              BEGIN
              H:=WR(DISK,H,SEX);
              Q:=0;
              END;
           IF M GTR 0 THEN N:=WR(DISK,N,B);
           M:=Q; N:=H;
           TRANSFER(SEX,0,B,0,C); % MOVE BACK TO B
           C:=0;
       END
   ELSE
   IF GT2=ARRAYDATA THEN
       BEGIN
       ARRAY DIMVECT[0:MAXBUFFSIZE];
       LSD:=T[1];
       IF H:=LSD.SPF=0 THEN % NULL VECTOR
       ELSE
           BEGIN
           T[1].INPTR:=N; T[1].DIMPTR:=M;
           C:=M×8;
           J:=CONTENTS(WS,LSD.DIMPTR,BUFFER); % DIM VECT
           PUTAWAY(C,J,WR,DISK,N,M,B,LEN); % STO DIM VECT
           J:=CONTENTS(WS,LSD.INPTR,DIMVECT);
           TRANSFER(DIMVECT,0,BUFFER,0,J);
           PUTAWAY(C,J,WR,DISK,N,M,B,LEN);
           J:=J-1;
           FOR LINE:=0 STEP 2 UNTIL J DO
              BEGIN
              PT:=GETFIELD(DIMVECT,LINE,2);
              STP:=CONTENTS(WS,PT,BUFFER);
              PUTAWAY(C,STP,WR,DISK,N,M,B,LEN);
              END;
           M:=C DIV 8; IF C MOD 8 NEQ 0 THEN M:=M+1; C:=0;
           IF M=WDSPERREC-1 THEN BEGIN N:=WR(DISK,N,B);
```

```
                        M:=0;                   END;
                    END;
                END;
                MOVETWO(T,DIR,K,WR,L,DISK);
            END;
        EOB:=1;
    IF M GTR 0 THEN N:=WR(DISK,N,B);
    IF K GTR 0 THEN L:=WR(DISK,L,DIR);
    LOCK(DISK);
    END;
BOOLEAN PROCEDURE LIBNAMES(A,B); REAL A,B;
BEGIN REAL T;
    A:=B:=GT1:=0;
%
%
    IF SCAN AND IDENT THEN
    BEGIN T←ACCUM[0]; T.[6:6]←"/";
    IF SCAN AND LOCKIT THEN GT1←1 ELSE IF IDENT THEN LIBNAMES←TRUE;
          A←T; B← JOBNUM;
    END
    ELSE LIBNAMES← TRUE;
    END;
PROCEDURE MESSAGEHANDLER;
    BEGIN
    LABEL ERR1;
%
    IF SCAN THEN IF IDENT THEN
    BEGIN INTEGER I; REAL R,S;
        PROCEDURE NOFILEPRESENT;
            BEGIN
            FILL BUFFER[*] WITH "FILE NOT"," ON DISK";
            FORMROW(3,0,BUFFER,0,16);
            END OF NOFILEPRESENT;
        PROCEDURE PRINTID(VARS); VALUE VARS; BOOLEAN VARS;
            BEGIN INTEGER I,J,K,L,M; ARRAY T[0:1]; BOOLEAN TOG;
            INTEGER NUM;
            J:=VARSIZE-1; M:=VARIABLES;
            FOR I:=0 STEP 1 UNTIL J DO
                BEGIN L:=CONTENTS(M,I,T); TOG:=GETFIELD(T,7,1)
                  =FUNCTION;
                IF NUM:=3×REAL(TOG AND VARS)+8+NUM GTR LINESIZE
                  THEN BEGIN TERPRINT; NUM:=3×REAL(TOG AND VARS)+8 END;
                IF VARS THEN
                    BEGIN FORMROW(0,1,T,0,7); L:=L+1;
                    IF TOG THEN FORMWD(0,"3(F)    ");
                    END ELSE
                IF TOG THEN BEGIN L:=L+1; FORMROW(0,1,T,0,7) END;
                END;
            IF L=0 THEN FORMWD(3,"6 NULL.") ELSE TERPRINT
            END;
    R:=ACCUM[0];
    FOR I:=0 STEP 1 UNTIL MAXMESS DO
        IF R=MESSTAB[I] THEN
            BEGIN R:=I; I:=MAXMESS+1
            END;
    IF I=MAXMESS+2 THEN
        CASE R OF
        BEGIN
        %  ------- SAVE -------
        IF NOT LIBNAMES(R,S) THEN
            IF NOT LIBRARIAN(R,S) THEN BEGIN
                SAVEWORKSPACE(R,S,GT1); %GT1 SET IN LIBNAMES
         GTA[0]←GTA[1]←0;TRANSFER(R,1,GTA,1,7);
                IF(GT1←SEARCHORD(LIBRARY,GTA, I ,7)) NEQ 0 THEN
                BEGIN GTA[0]←GTA[1]←0;TRANSFER(R,1,GTA,1,7);
                      STOREORD(LIBRARY,GTA,I+(IF GT1=1 THEN -1 ELSE 1));
                END; LIBSIZE←LIBSIZE+1;
        END
            ELSE
                BEGIN
                FILL BUFFER[*] WITH "FILE ALR","EADY ON ",
                    "DISK    ";
                FORMROW(3,0,BUFFER,0,20);
                END
        ELSE GO ERR1;
        %  ------- LOAD -------
        IF NOT LIBNAMES(R,S) AND R NEQ 0 THEN
            IF LIBRARIAN(R,S) THEN
                BEGIN ARRAY A[0:1];
                LOADWORKSPACE(R,S,A);
                END
            ELSE NOFILEPRESENT
    ELSE GO ERR1;
    %  ------- DROP -------
    IF CURRENTMODE=CALCMODE THEN
    IF NOT LIBNAMES(R,S) THEN
        IF LIBRARIAN(R,S) THEN
        BEGIN FILE ELIF DISK (1,1);
        FILL ELIF WITH R,S; WRITE(ELIF[0]);
        CLOSE(ELIF,PURGE)
       ;GTA[0]←GTA[1]←0;TRANSFER(R,1,G[A,1,7]);
        IF SEARCHORD(LIBRARY,GTA,I,7)=0 THEN DELETE1(LIBRARY,I);
        LIBSIZE←LIBSIZE-1;
        END
        ELSE NOFILEPRESENT
    ELSE
    IF NOT BOOLEAN(SUSPENSION)THEN PURGEWORKSPACE(WORKSPACE)
    ELSE GO ERR1 ELSE GO ERR1;
    %  ------- COPY -------
    IF LIBNAMES(R,S) THEN
        IF LIBRARIAN(R,S) THEN
        LOADWORKSPACE(R,S,ACCUM)
    ELSE NOFILEPRESENT
    ELSE GO ERR1;

    %  --------  VARS -------
    PRINTID(TRUE);

    %------- FNS -------
    PRINTID(FALSE);
    %-------- LOGGED ---------------
;
    %-------- MSG --------
    ERRORMESS(SYNTAXERROR,LADDRESS,0);
    %-----WIDTH (INTEGER) -----------------------------
    IF NOT SCAN THEN BEGIN NUMBERCON(LINESIZE, ACCUM);
                     FORMROW(3,0,ACCUM,2,ACOUNT); END
        ELSE IF NUMERIC AND I:=ACCUM[0] GTR 9 AND I LEQ 72
          THEN BEGIN TERPRINT; LINESIZE:=I; STOREPSR;
        END
    %IF A NUMBER CONVERSION ERROR, RESULT WILL BE ZERO
      %AND WE"LL GET AN ERROR ANYWAY
        ELSE GO TO ERR1;
    %-------- OPR --------
    ;
    %------DIGITS (INTEGER) ----------------------
    IF NOT SCAN THEN BEGIN NUMBERCON(DIGITS,ACCUM);
                           FORMROW(3,0,ACCUM,2,ACOUNT); END
        ELSE IF NUMERIC AND I:=ACCUM[0] GEQ 0 AND I LEQ 12
          AND ERR=0 THEN BEGIN DIGITS:=I; STOREPSR END
    ELSE GO TO ERR1;
    %-------- OFF --------
    BEGIN
    IF SCAN THEN IF ACCUM[0]="7DISCAR" THEN
        ELIMWORKSPACE(WORKSPACE) ELSE
        GO TO ERR1;
   FILL ACCUM[*] WITH "END OF R","UN      ";
    FORMROW(3,MARGINSIZE,ACCUM,0,10);
    CURRENTMODE:=CALCMODE;
   GT1:=CSTATION;
   CSTATION:=GT1&0[CAPLOGGED]
  ;GO TO FINIS;
    END;
    %--------ORIGIN----------------------------------
    IF NOT SCAN THEN BEGIN NUMBERCON(ORIGIN,ACCUM);
                           FORMROW(3,0,ACCUM,2,ACOUNT) END
    ELSE IF NUMERIC AND ERR=0 THEN BEGIN ORIGIN:=
    I:=ACCUM[0]; STOREPSR END ELSE GO TO ERR1;
    %--------SEED---------------------------------
    IF NOT SCAN THEN BEGIN NUMBERCON(SEED,ACCUM);
                           FORMROW(3,0,ACCUM,2,ACOUNT) END
    ELSE IF NUMERIC AND ERR=0 THEN BEGIN
        SEED:=ABS(I:=ACCUM[0]);
        STOREPSR END ELSE GO TO ERR1;
    %--------FUZZ------------------------------------
    IF NOT SCAN THEN BEGIN
        NUMBERCON(FUZZ,ACCUM);
        FORMROW(3,0,ACCUM,2,ACOUNT) END
    ELSE IF NUMERIC AND ERR=0 THEN BEGIN FUZZ:=ABS(ACCUM[0]);
        STOREPSR END ELSE GO TO ERR1;
    %------- SYN, NOSYN---------------------------------
   NOSYNTAX:=0; NOSYNTAX:=1;
    %----------------STORE--------------------------
    IF SYMBASE NEQ 0 THEN PROCESS(WRITEBACK);
```

```
        %-----------------ABORT-------------------------
        BEGIN IF BOOLEAN(SUSPENSION) THEN
            SP[0,0]:=0; NROWS:=-1;
%%%
        SUSPENSION:=0;
        STOREPSR
        END;
        %-----------------SI-----------------------------
        IF BOOLEAN(SUSPENSION) THEN
            BEGIN GT1:=0;
            PROCESS(LOOKATSTACK);
        END ELSE FORMWD(3,"6 NULL.");
        %-----------------SIV----------------------------
        IF BOOLEAN(SUSPENSION) THEN
            BEGIN GT1:=1;
            PROCESS(LOOKATSTACK);
        END ELSE FORMWD(3,"6 NULL.");
        %-----------------ERASE--------------------------
        IF CURRENTMODE=FUNCMODE OR BOOLEAN(SUSPENSION) THEN GO TO ERR1
        ELSE WHILE SCAN AND IDENT DO
            BEGIN % LOOK FOR THE IDENTIFIER NAME IN ACCUM
            TRANSFER(ACCUM,2,GTA,0,7);
            IF (IF VARIABLES=0 THEN FALSE ELSE
                SEARCHORD(VARIABLES,GTA,GT1,7)=0) THEN
                BEGIN  % FOUND A SYMBOL TABLE ENTRY MATCHING NAME
                DELETE1(VARIABLES,GT1); % REMOVE FROM SYMBOLTABLE
                IF VARSIZE:=VARSIZE-1=0 THEN VARIABLES:=0;
                COMMENT IF NOTHING IS IN THE UNIT IT IS DELETED;

                % CHECK IF THERE IS MORE TO DELETE
                IF GT1:=GETFIELD(GTA,7,1)=FUNCTION THEN
                    BEGIN
                    RELEASEUNIT(GETFIELD(GTA,FPTF,FFL));
                    RELEASEUNIT(GETFIELD(GTA,FSQF,FFL));
                    END
                ELSE IF GT1 GTR 0 THEN % MUST BE AN ARRAY
                RELEASEARRAY(GTA[1]);
                END ELSE % THERE IS NO SUCH VARIABLE
                ERRORMESS(LABELERROR,LADDRESS,0);
            END;  % OF TAKING CARE OF ERASE
        %------------ ASSIGN ----------------------------
|
        %------------ DELETE ----------------------------
|
        %------------ LIST ------------------------------
|
        % -------------DEBUG -----------------------------
        IF SCAN AND IDENT THEN
            IF ACCUM[0]="6POLISH" THEN POLBUG:=ABS(POLBUG-1);

    %-------------------------- FILES ---------------------
    IF LIBSIZE>1 THEN
    BEGIN FOR I←1 STEP 1 UNTIL LIBSIZE-1 DO
            BEGIN R←CONTENTS(LIBRARY,I   ,ACCUM);
                FORMROW(0,1,ACCUM,2,6);
            END; TERPRINT;
    END ELSE FORMWD(3,"6 NULL.");
    %---------------------- END OF CASES ----------------------------
        END ELSE GO TO ERR1;
    IF CURRENTMODE=FUNCMODE THEN INDENT(-CURLINE);
    END ELSE
    IF QUOTE THEN EDITLINE ELSE
    ERR1:  ERRORMESS(SYNTAXERROR,0,0);
    INDENT(0);
    TERPRINT;
    END;
EAL PROCEDURE LINENUMBER(R); VALUE R; REAL R;
    BEGIN
    REAL STREAM PROCEDURE CON(R); VALUE R;
        BEGIN SI:=LOC R; DI:=LOC CON; DS:=8DEC
        END;
    LINENUMBER:=CON( ENTIER( (R+.00005)×10000))
    END;
EFINE DELIM="""#, ENDCHR="$"#;
OOLEAN PROCEDURE WITHINALINE(COMMAND,OLD,NEW,CHAR,WORD);
 VALUE COMMAND,CHAR,WORD; INTEGER COMMAND,CHAR,WORD;
 ARRAY OLD, NEW[0]; BEGIN
OOLEAN STREAM PROCEDURE WITHINLINE(COMMAND,OLD,NEW,CHAR,WORD);
    VALUE COMMAND,CHAR,WORD;
    BEGIN
    LOCAL OLDLINE,NEWLINE,F,BCHR;
    LOCAL N,M,T;
```

80

```
      LOCAL X,Y,7;
      LABEL LOOKING,FOUND,BETWEEN,TAIL,FINISH,
      OVER;
      DI:=NEW; WORD(DS:=8LIT" ");
      SI:=LOC CHAR; DI:=LOC T; DI:=DI+1; DS:=7CHR;
      SI:=COMMAND;
      TALLY:=T; X:=TALLY; TALLY:=2; Y:=TALLY; TALLY:=32; Z:=TALLY;
      TALLY:=0;
          IF SC≠"←" THEN
          BEGIN BCHR:=SI; SI:=OLD; OLDLINE:=SI;
          DI:=NEW; NEWLINE:=DI; SI:=BCHR;
          63(IF SC=DELIM THEN JUMP OUT ELSE SI:=SI+1; TALLY
          :=TALLY+1); N:=TALLY;
          IF TOGGLE THEN
              BEGIN
              SI:=SI+1; TALLY:=0;
              63(IF SC=DELIM THEN TALLY:=0 ELSE
                  IF SC="←" THEN JUMP OUT ELSE TALLY:=TALLY+1; SI:=SI+1);
              IF TOGGLE THEN M:=TALLY;
              DI:=OLDLINE; SI:=BCHR;
              2( X( Y( Z( CI:=CI+F;
              GO LOOKING; GO FOUND; GO BETWEEN; GO TAIL; GO FINISH;
 LOOKING: %************ LOOKING FOR THE FIRST UNIQUE STRING**************
         IF SC=DELIM THEN BEGIN SI:=SI+1; TALLY:=2; F:=TALLY ;
            DI:=NEWLINE; GO BETWEEN END ELSE
         IF N SC=DC THEN BEGIN SI:=OLDLINE; SI:=SI+N; OLDLINE:=SI;
            DI:=NEWLINE; SI:=BCHR; TALLY:=1; F:=TALLY;
            GO FOUND END ELSE
                BEGIN SI:=OLDLINE; DI:=NEWLINE; DS:=CHR; NEWLINE:=DI;
                OLDLINE:=SI; SI:=BCHR;   DI:=OLDLINE
                END; GO OVER;
 FOUND:   %**************FOUND THE FIRST UNIQUE STRING *****************
         IF SC=DELIM THEN BEGIN SI:=SI+1; TALLY:=2;
            F:=TALLY; GO BETWEEN END ELSE
         DS:=CHR;  GO OVER;
 BETWEEN:  % ********** BETWEEN THE // ***********************************
         IF SC=DELIM THEN BEGIN SI:=SI+1; NEWLINE:=DI; DI:=OLDLINE;
            TALLY:=3; F:=TALLY; GO TAIL END ELSE
         IF SC="←" THEN BEGIN TALLY:=4; F:=TALLY;
                SI:=OLDLINE; GO FINISH END ELSE
         DS:=CHR; GO OVER;
 TAIL:  % ******* THE TAIL END OF THE COMMAND ****************************
         IF M SC=DC THEN BEGIN DI:=NEWLINE; SI:=OLDLINE; TALLY:=4;
                F:=TALLY; GO FINISH END ELSE
            BEGIN SI:=SI-M; DI:=DI-M; DI:=DI+1; OLDLINE:=DI; END;
         GO OVER;
 FINISH:   % ********FINISH UP THE CHR MOVE FROM THE OLD TO NEW*********
              DS:=CHR; OVER:)));
              TALLY:=CHAR; X:=TALLY; TALLY:=1; Y:=TALLY;
              Z:=TALLY);
              SI:=NEW; DI:=OLD; DS:=WORD WDS; TALLY:=1;
              WITHINLINE:=TALLY
              END
         END
     END OF WITHINALINE;
   WITHINALINE := WITHINLINE(COMMAND,OLD,NEW,CHAR,WORD);
   END OF PHONY WITHINALINE;
 PROCEDURE EDITLINE;
     BEGIN ARRAY T[0:MAXBUFFSIZE];
     INITBUFF(T,BUFFSIZE);
     TRANSFER(OLDBUFFER,0,T,0,LENGTH(OLDBUFFER,TRUE));
     IF WITHINALINE(ADDRESS,T,OLDBUFFER,BUFFSIZEx8,BUFFSIZE) THEN
         BEGIN MOVEWDS(OLDBUFFER,BUFFSIZE,BUFFER);

         IF SCAN AND RGTPAREN THEN
            ERRORMESS(SYNTAXERROR,LADDRESS,0) ELSE EDITOG:=1;
         END;


         FORMROW(3,0,BUFFER,0,LENGTH(BUFFER,FALSE));
     END;
 PROCEDURE CHECKSEQ(SEQ,L,INC); REAL SEQ,L,INC;
     BEGIN
     INTEGER I,J;
     I:=Lx10000 MOD 10000;
     FOR J:=-4 STEP 1 WHILE J LSS 0 AND I MOD 10=0 DO
         I:=I/10;
     INC:=10*J;
     SEQ:=L;
     END;
 PROCEDURE FUNCTIONHANDLER;
     BEGIN
     LABEL ENDHANDLER;
```

```
    OWN BOOLEAN EDITMODE;
    DEFINE FPT=FUNCPOINTER#,
        FSQ=FUNCSEQ#,
        SEQ=CURLINE#,
        INC=INCREMENT#,
    MODE=SPECMODE#,
        ENDDEFINES=#;
    INTEGER STREAM PROCEDURE DELPRESENT(ADDR); VALUE ADDR;
        BEGIN LABEL L,FINIS;
        LOCAL Q;
        DI:=LOC Q; DS:=RESET; DS:=5SET; DS:=2RESET; DS:=2SET;
        % LEFT-ARROW / QUESTION MARK
        SI:=ADDR;
        L: DI:=LOC Q;
        IF SC=DELCHR THEN
            BEGIN ADDR:=SI; SI:=LOC Q; DI:=ADDR; DS:=LIT" ";
            TALLY:=1; DELPRESENT:=TALLY; GO TO FINIS;
            END;
        IF SC=DC THEN GO TO FINIS; SI:=SI-1;
        IF SC=DC THEN GO TO FINIS;
        GO TO L;
        FINIS:
        END;
INTEGER PROCEDURE OLDLABCONFLICT(PT,S); VALUE PT,S;
    INTEGER PT; REAL S;
    IF PT NEQ 0 THEN
        BEGIN INTEGER K; ARRAY L[0:1];
        ADDRESS:=ABSOLUTEADDRESS;
        WHILE LABELSCAN(L,0) AND ERR EQL 0 DO
        IF SEARCHORD(PT,L,K,8)=0 THEN
            IF L[1] NEQ S THEN ERR:=24;
    OLDLABCONFLICT:=ERR
    END;
INTEGER PROCEDURE ELIMOLDLINE(PT,SQ,L); VALUE PT,SQ,L; INTEGER PT,
        SQ,L; FORWARD;
INTEGER PROCEDURE STOREAWAY(PT,SQ,B,SEQ); VALUE SEQ;
    INTEGER PT,SQ; REAL SEQ; ARRAY B[0]; FORWARD;
    PROCEDURE BUFFERCLEAN(BUFFER,BUFFSIZE,ADDR); VALUE BUFFSIZE,
       ADDR; REAL ADDR; INTEGER BUFFSIZE; ARRAY BUFFER[0];
       FORWARD; COMMENT THIS IS A PHONEY DEAL, BUT I CAN"T
      DECLARE CLEANBUFFER FORWARD (MOVE IT UP HERE LATER);
PROCEDURE EDITDRIVER(PT,SQ,I,K); VALUE PT,SQ,I,K;
    INTEGER PT,SQ,I,K;
    BEGIN ARRAY C,LAB[0:1],OLD,NEW[0:MAXBUFFSIZE];
    STREAM PROCEDURE BL(A);
        BEGIN DI:=A; MAXBUFFSIZE(DS:=8LIT" ") END;
    DEFINE MOVE=MOVEWDS#;
    REAL T,SEQ; INTEGER A,B,L,M;
    T:=ADDRESS:
    FOR A:=I STEP 1 WHILE A LEQ K AND EDITMODE DO
        BEGIN B:=CONTENTS(PT,A,C); BL(OLD);
        SEQ:=C[0];
        B:=CONTENTS(SQ,C[1],OLD);
        IF EDITMODE:=WITHINALINE(T,OLD,NEW,BUFFSIZE×8,BUFFSIZE)
        THEN BEGIN MOVE(BUFFER,MAXBUFFSIZE+1,NEW);
            MOVE(OLD,MAXBUFFSIZE,BUFFER);
            IF EDITMODE:=ERR:=OLDLABCONFLICT(PT,C[0])=0 THEN
                BEGIN B:=ELIMOLDLINE(PT,SQ,C[1]);
                DELTOG:=DELPRESENT(ADDRESS);
                DELETE1(SQ,C[1]); DELETE1(PT,A+B); C[1]:=
                STORESEQ(SQ,BUFFER,LENGTH(BUFFER,FALSE));
                STOREORD(PT,C,A+B);
                RESCANLINE; L:=0; M:=1; LAB[1]:=C[0];
                WHILE LABELSCAN(C,0) DO
                    BEGIN MOVEWDS(C,1,LAB);
                    IF(IF FUNCSIZE=0 THEN TRUE ELSE L:=
                    SEARCHORD(PT,C,M,8)NEQ 0) THEN
                        BEGIN B:=B+1; FUNCSIZE:=FUNCSIZE+1;
                        STOREORD(PT,LAB,L+M-1)
               END END;
           A:=A+B; K:=K+B;
            COMMENT THE NEXT LINE CAUSED A SYSTEM CRASH AFTER THE EDIT
           IF NOSYNTAX=0 THEN PROCESS(XEQUTE);
           END END;
       MOVE(NEW,MAXBUFFSIZE+1,BUFFER)
   END END;
    PROCEDURE LISTLINE(PT,SQ,I); VALUE PT,SQ,I; INTEGER PT,SQ,I;
        BEGIN
        GT1:=CONTENTS(PT,I,GTA);
        INDENT(GTA[0]);
        GT1:=CONTENTS(SQ,GTA[1],BUFFER);
        CHRCOUNT:=CHRCOUNT-1;
        FORMROW(1,0,BUFFER,0,GT1);
```

```
         END;
 INTEGER PROCEDURE DISPLAY(A,B,PT,SQ); VALUE A,B,PT,SQ;
     INTEGER PT,SQ; REAL A,B;
     IF A LEQ B AND FUNCSIZE NEQ 0 THEN
         BEGIN
         ARRAY C[0:1];
         INTEGER I,J,K;
         DEFINE CLEANBUFFER=BUFFERCLEAN#;
         A:=LINENUMBER(A); B:=LINENUMBER(B);
         C[0]:=A;
         I:=SEARCHORD(PT,C,K,8);
         I:=(IF I=2 THEN IF K LSS FUNCSIZE-1 THEN K:=K+1 ELSE
              K ELSE K);
         IF A NEQ B THEN
             BEGIN
             C[0]:=B; B:=SEARCHORD(PT,C,K,8);
             END;
         IF EDITMODE THEN % MAY HAVE ONLY ONE LINE TO EDIT
             IF I=K THEN
                 IF A NEQ 0 THEN %NOT EDITING THE HEADER
                     EDITDRIVER(PT,SQ,I,K)
                 ELSE %EDITING THE FUNCTION HEADER, FIX LATER.
                     ERR:=31
                 ELSE %EDITING MORE THAN ONE LINE
             BEGIN MODE:=EDITING;
                 IF A=0 THEN I:=I+1;
             CLEANBUFFER(BUFFER,BUFFSIZE,ADDRESS);
             MOVE(BUFFER,BUFFSIZE,OLDBUFFER);
             LOWER:=I; UPPER:=K
             END
        ELSE %NOT EDITING, MUST BE A LIST
        BEGIN
        FORMWD(3,"1       ");
        IF K=I THEN % LISTING A SINGLE LINE
            BEGIN LISTLINE(PT,SQ,I);
            FORMWD(3,"1       ");
            END ELSE % LISTING A SET OF LINES
            BEGIN MODE:=DISPLAYING;
            LOWER:=I; UPPER:=K
            END;
        END;
        EOB:=1;
        END ELSE DISPLAY:=20;
INTEGER PROCEDURE DELETE(A,B,PT,SQ); VALUE A,B;
    INTEGER PT,SQ; REAL A,B;
    IF A LEQ B AND FUNCSIZE NEQ 0 AND A NEQ 0 THEN
        BEGIN
        INTEGER I,J,K,L;
        ARRAY C[0:1];
        A:=LINENUMBER(A);
        B:=LINENUMBER(B);
        C[0]:=A;
        IF SEARCHORD(PT,C,K,8)=2 THEN K:=K+1;
        C[0]:=B;
        IF SEARCHORD(PT,C,I,8)=1 THEN I:=I-1;
        IF K GTR I OR I GEQ FUNCSIZE THEN DELETE:=21 ELSE
            BEGIN
            FOR J:=K STEP 1 UNTIL I DO
                BEGIN A:=CONTENTS(PT,J,C);
                L:=ELIMOLDLINE(PT,SQ,C[1]);
                FUNCSIZE:=FUNCSIZE+L; I:=I+L; K:=K+L; J:=J+L;
                DELETE1(SQ,C[1])
                END;
            FUNCSIZE:=FUNCSIZE-(I-K+1)
            ; EOB:=1;
            DELETEN(PT,K,I);
        IF FUNCSIZE=0 THEN
             BEGIN
            PT:=0; RELEASEUNIT(SQ); SQ:=0;
            STOREPSR;
            END;
        END;
        END ELSE DELETE:=22;
   INTEGER PROCEDURE ELIMOLDLINE(PT,SQ,L); VALUE PT,SQ,L;
   INTEGER PT,SQ,L;
        BEGIN INTEGER K,J;
        REAL AD;
        ARRAY T[0:MAXBUFFSIZE],LAB[0:1];
         AD:=ADDRESS;
        MOVEWDS(BUFFER,MAXBUFFSIZE+1,T);
        INITBUFF(BUFFER,BUFFSIZE);
        K:=CONTENTS(SQ,L,BUFFER);
        RESCANLINE;
```

```
       WHILE LABELSCAN(LAB,0) DO
            IF SEARCHORD(PT,LAB,K,8)=0 THEN
            BEGIN DELETE1(PT,K); J:=J-1 END;
       ADDRESS:=AD;
       MOVEWDS(T,MAXBUFFSIZE+1,BUFFER);
       ELIMOLDLINE:=J
       END;
INTEGER PROCEDURE STOREAWAY(PT,SQ,B,SEQ); VALUE SEQ;
       INTEGER PT,SQ; REAL SEQ; ARRAY B[0];
       BEGIN DEFINE BUFFER=B#;
       ARRAY C,LAB[0:1];
       INTEGER I,J,K,L;
       BOOLEAN TOG;
       SEQ:=LINENUMBER(SEQ);
       C[0]:=SEQ;
       IF TOG:=(PT=0 OR FUNCSIZE=0) THEN
            BEGIN SEQUENTIAL(SQ:=NEXTUNIT); I:=0
            END ELSE
       IF J:=SEARCHORD(PT,C,I,8)=0 THEN
            BEGIN
            K:=ELIMOLDLINE(PT,SQ,C[1]);
            I:=I+K; FUNCSIZE:=FUNCSIZE+K;
            DELETE1(PT,I);
            FUNCSIZE:=FUNCSIZE-1;
            DELETE1(SQ,C[1]);
            END ELSE
       I:=I+J-1;
       RESCANLINE;
       DELTOG:=DELPRESENT(ADDRESS);
       K:=STORESEQ(SQ,BUFFER,LENGTH(BUFFER,TRUE));
       LAB[1]:=SEQ; L:=0; J:=1;
       IF TOG THEN PT:=NEXTUNIT;
       WHILE LABELSCAN(C,0) DO
            BEGIN
            MOVEWDS(C,1,LAB);
            IF (IF FUNCSIZE=0 THEN TRUE ELSE L:=
                 SEARCHORD(PT,C,J,8)NEQ 0 ) THEN
                 BEGIN I:=I+1; FUNCSIZE:=FUNCSIZE+1;
                 STOREORD(PT,LAB,L+J-1);
                 END
            END;
       C[1]:=K;
       C[0]:=SEQ;
       FUNCSIZE:=FUNCSIZE+1;
       STOREORD(PT,C,I);
       IF TOG THEN STOREPSR;
       EOB:=1;
       END;
    BOOLEAN PROCEDURE BOUND(PT); VALUE PT; INTEGER PT;
    IF NOT(BOUND:=NUMERIC) THEN
       IF IDENT AND FUNCSIZE GTR 0 THEN
       BEGIN ARRAY L[0:1]; INTEGER K;
       REAL T,U;
       REAL STREAM PROCEDURE CON(A);
       VALUE A;
            BEGIN SI:=LOC A; DI:=LOC CON; DS:=8OCT
            END;
       TRANSFER(ACCUM,2,L,1,7);
       IF BOUND:=SEARCHORD(PT,L,K,8)=0 THEN
            BEGIN T:=ADDRESS;
            U:=CON(MAX(L[1],0))/10000; %ARGS AND RESULT ARE NEG
            IF SCAN AND PLUS OR MINUS THEN
                 BEGIN K:=(IF PLUS THEN 1 ELSE -1);
                 IF SCAN AND NUMERIC THEN
                 ACCUM[0]:=MAX(U+K×ACCUM[0],0) ELSE
                      BEGIN ACCUM[0]:=U;
                      ADDRESS:=T;
                      END;
                 END ELSE BEGIN ACCUM[0]:=U; ADDRESS:=T
                          END;
            EOB:=0;
            END
       END;


    PROCEDURE FINISHUP;
    BEGIN COMMENT GET HIM BACK TO CALCULATOR MODE;
    IF FUNCPOINTER=0 THEN % HE DELETED EVERY THING
       BEGIN TRANSFER(PSR,FSTART×8,GTA,0,8);
       IF SEARCHORD(VARIABLES,GTA,GT1,7)=0 THEN
            BEGIN DELETE1(VARIABLES,GT1);
            IF VARSIZE:=VARSIZE-1=0 THEN VARIABLES:=0;
            END ELSE SPOUT(9198260);
```

```
        END;
    DELTOG:=CURRENTMODE:=CURLINE:=INCREMENT:=0;
    STOREPSR;
    END;

    LABEL SHORTCUT;
    REAL L,U,TADD;
    STREAM PROCEDURE CLEANBUFFER(BUFFER,BUFFSIZE,ADDR);
    VALUE BUFFSIZE,ADDR;
    BEGIN LABEL L; LOCAL T,U,TSI,TDI;
    SI:=ADDR; SI:=SI-1; L:
    IF SC NEQ "]" THEN
        BEGIN SI:=SI-1; GO TO L END;
    SI:=SI+1; DI:=LOC T; SKIP 2 DB; DS:=2SET;
    DI:=BUFFER; TDI:=DI; DI:=LOC T; TSI:=SI;
    BUFFSIZE(8(IF TOGGLE THEN DS:=LIT" " ELSE
        IF SC=DC THEN
            BEGIN SI:=LOC U; DI:=TDI; DS:=LIT" "
            END ELSE
        BEGIN TSI:=SI; SI:=SI-1; DI:=LOC U; DS:=CHR;
        DI:=TDI; SI:=LOC U; DS:=CHR; TDI:=DI; DI:=LOC T;
        SI:=TSI
        END))
    END;
    PROCEDURE BUFFERCLEAN(BUFFER,BUFFSIZE,ADDR); VALUE BUFFSIZE,
       ADDR; REAL ADDR; INTEGER BUFFSIZE; ARRAY BUFFER[0];
       CLEANBUFFER(BUFFER,BUFFSIZE,ADDR);
COMMENT DETERMINE WHETHER OR NOT WE CAME FROM CALCULATOR MODE;
    ERR:=0;
    IF BOOLEAN(SUSPENSION) THEN GO TO ENDHANDLER;
    BEGIN DEFINE STARTSEGMENT=#; %/////////////////////////////////
    IF GT1:=CURRENTMODE=CALCMODE THEN % TAKE CARE OF HEADER.
        BEGIN ARRAY A[0:MAXHEADERARGS];
    LABEL HEADERSTORE,FORGETITFELLA;
        IF FUNCTIONHEADER(A,TADD) THEN %HEADER OK
            IF VARIABLES NEQ 0 THEN % MAY BE A RE-DEFINITION
                BEGIN COMMENT GET THE FUNCTION NAME;
                TRANSFER(A,1,GTA,0,7);
                IF GT2:=SEARCHORD(VARIABLES,GTA,GT3,7)=0 THEN
                    COMMENT RE-DEFINING A FUNCTION.  MAKE SURE NULL ;
                    IF GETFIELD(GTA,7,1)=FUNCTION AND
                    (A[1]+A[2]+A[3])=0  THEN %NULL HEADER--OK
%--------------------SET UP FOR CONTINUATION OF DEFINITION------
    BEGIN
    FUNCPOINTER:=GETFIELD(GTA,FPTF,FFL);
    FUNCSEQ:=GETFIELD(GTA,FSQF,FFL);
    GT3:=CURLINE:=TOPLINE(FPT);
    CHECKSEQ(CURLINE,GT3,INC); %SET THE INCREMENT
    COMMENT THE CURRENTLINE IS SET TO THE LAST LINE OF THE
    FUNCTION;
    FUNCSIZE:=SIZE(FPT);
    CURLINE:=CURLINE+INC;
    DELTOG:=DELPRESENT(ADDRESS);
    END ELSE
%------------------REDEFINING THE HEADER OF A DEFINED FUNCTION----
                    GO TO FORGETITFELLA
                    ELSE
%--------------------NAME NOT FOUND IN THE DIRECTORY, SET UP
HEADERSTORE:
    BEGIN COMMENT GET THE HEADER TO INSERT AT LINE 0;
    ARRAY OLDBUFFER[0:MAXBUFFSIZE];
    INTEGER L,U,F,K,J;
    INTEGER A1,A2;
        COMMENT FUNCTIONHEADER RETURN AN ARRAY WITH THE
        FOLLOWING VALUES:
        A[0]   = FUNCTION NAME , I.E., 0AAAAAAA
        A[1]   = 0 IF NO RESULT, 1 IF A RESULT IS RETURNED BY THE
                 FUNCTION.
        A[2]   = NUMBER OF ARGUMENTS TO THE FUNCTION.
        A[3]   = NUMBER OF LOCALS + RESULT +ARGUMENTS.
        A[4],...A[N]   ARE ALL OF THE LOCALS, RESULT, AND ARGUMENTS.
        THE RESULT IS FIRST, THEN THE SECOND ARGUMENT, THEN
        THE FIRST ARGUMENT, FOLLOWED BY THE LOCALS.  ALL
        ARE OF THE FORM 0XXXXXXX;
        U:=(A1:=A[1])+(A2:=A[2])+3;
        FOR L:=4 STEP 1 UNTIL U DO %LOOK FOR DUPLICATES AMONG
           FOR K:=L+1 STEP 1 UNTIL U DO %THE RESULT/ARGUMENT SET
              IF A[L]=A[K] THEN GO TO FORGETITFELLA;
    SEQUENTIAL(FUNCSEQ:=NEXTUNIT);
    SETFIELD(GTA,8,8,STORESEQ(FUNCSEQ,OLDBUFFER,
          HEADER(TADD.[1:23],TADD.[24:24],OLDBUFFER)));
    SETFIELD(GTA,0,8,0);
    STOREORD(F:=FUNCPOINTER:=NEXTUNIT,GTA,0);
```

```
SETFIELD(GTA,0,8,0); SETFIELD(GTA,8,8,0);
FOR L:=4 STEP 1 UNTIL U DO
    BEGIN GTA[0]:=A[L]; IF A1 GTR 0 THEN
         BEGIN A1:=0; GTA[1]:=-1; %"RESULT" SET TO -1
         STOREORD(F,GTA,0);
         END ELSE %LOOKING AT THE ARGUMENTS
         BEGIN K:=SEARCHORD(F,GTA,J,8);
         GTA[1]:=A2-4; A2:=A2-1; GTA[0]:=A[L];
         STOREORD(F,GTA,J+K-1);
    END END;
FUNCSIZE:=U:=U-2; U:=A[3]-U+L;
FOR L:=L STEP 1 UNTIL U DO %GET LOCALS INTO THE LABEL TABLE
    BEGIN GTA[0]:=A[L];
    IF K:=SEARCHORD(F,GTA,J,8) NEQ 0 THEN %NOT YET IN TABLE.
         BEGIN GTA[0]:=A[L]; GTA[1]:=0;
         STOREORD(F,GTA,J+K-1);
         FUNCSIZE:=FUNCSIZE+1
         END;
    END;
    GTA[1]:=0&ENTIER(A[1])[CRETURN]&ENTIER(A[2])[CNUMBERARGS];
CURLINE:=INCREMENT:=1;
DELTOG:=0;
COMMENT GET THE "TYPE" OF THE FUNCTION LATER WHEN THERE
IS A PLACE FOR IT.  THE TYPE IS EITHER 1 (FUNCTION CALL), OR
0 (SUBROUTINE CALL);
END
%-----------------------------------------------------
                END ELSE % VARAIBLES=0, MAKE UP A DIRECTORY
                BEGIN GT3:=0; GT2:=1; GO TO HEADERSTORE
                END
            ELSE % HEADER SYNTAX IS BAD
            GO TO ENDHANDLER;
    COMMENT WE MAKE IT TO HERE IF ALL IS WELL ABOVE;
        IF GT2 NEQ 0 THEN %NAME NOT FOUND IN DIRECTORY;
            BEGIN
            TRANSFER(A,1,GTA,0,7); %GET FUNCTION NAME
            SETFIELD(GTA,7,1,FUNCTION);
            SETFIELD(GTA,FPTF,FFL,FUNCPOINTER);
            SETFIELD(GTA,FSQF,FFL,FUNCSEQ);
            IF VARIABLES=0 THEN
                VARIABLES:=NEXTUNIT;
            STOREORD(VARIABLES,GTA,GT3+GT2-1);
            VARSIZE:=VARSIZE+1;
            END;
            CURRENTMODE:=FUNCMODE;
            TRANSFER(GTA,0,PSR,FSTART×8,8);
            STOREPSR;
        IF SCAN THEN GO TO SHORTCUT;
        IF FALSE THEN
        FORGETTTFELLA: ERRORMESS(ERR:=LABELERROR,TADD.[1:23],0);
        END ELSE % WE ARE IN FUNCTION DEFINITION MODE
    IF GT1:=MODE NEQ 0 THEN % A SPECIAL FUNCTION SUCH AS DISPLAY OR EDIT
        BEGIN L:=LOWER;
        IF GT1=DISPLAYING THEN
            LISTLINE(FPT,FSQ,L) ELSE
        IF GT1=EDITING THEN
            BEGIN INITBUFF(BUFFER,BUFFSIZE);
            MOVE(OLDBUFFER,BUFFSIZE,BUFFER);
            EDITMODE:=TRUE; ADDRESS:=ABSOLUTEADDRESS;
            EDITDRIVER(FPT,FSQ,L,L)
            ;IF NOT EDITMODE THEN
                BEGIN MODE:=0; ERR:=30
                END;
        END ELSE
        IF GT1=RESEQUENCING THEN
            IF GT1:=L LEQ UPPER THEN
                BEGIN GT2:=CONTENTS(FPT,L,GTA);
                GT3:=GTA[0]:=LINENUMBER(CURLINE);
                DELETE1(FPT,L);
                STOREORD(FPT,GTA,L);
                CURLINE:=CURLINE+INCREMENT;
                GT2:=CONTENTS(FSQ,GTA[1],BUFFER); RESCANLINE;
                WHILE (IF ERR NEQ 0 THEN FALSE ELSE
                    LABELSCAN(GTA,0)) DO
                    IF GT1:=SEARCHORD(FPT,GTA,GT2,8)=0 THEN
                    BEGIN GTA[1]:=GT3; DELETE1(FPT,GT2);
                    STOREORD(FPT,GTA,GT2)
                    END ELSE ERR:=16
                END
            ELSE MODE:=0;
        LOWER:=L+1;
        IF LOWER GTR UPPER THEN
            BEGIN IF MODE=DISPLAYING THEN
            FORMWD(3,"1       ");                                        09231400
            MODE:=0;                                                     09231500
           END;                                                          09231600
        GO TO ENDHANDLER                                                 09231700
        END;                                                             09231800
    END ; %OF BLOCK STARTED ON LINE 9225115 ////////////////////         09232000
                                                                         09233000
                                                                         09234000
                                                                         09235000
    IF ERR=0 AND EOB=0 THEN                                              09236000
                                                                         09237000
SHORTCUT:  BEGIN LABEL RGTBRACK,DELOPTION; %///////////////////////      09238000
   IF DELV THEN FINISHUP ELSE                                            09239000
      IF LFTBRACKET THEN                                                 09240000
        BEGIN                                                            09241000
        IF SCAN THEN                                                     09242000
           IF BOUND(FPT) THEN                                            09243000
            BEGIN L:=ACCUM[0];                                           09244000
            IF SCAN THEN                                                 09245000
               IF QUADV OR EDITMODE:=(QUOTEQUAD) THEN                    09246000
                IF SCAN THEN                                             09247000
                   IF BOUND(FPT) THEN                                    09248000
                    BEGIN U:=ACCUM[0];                                   09249000
RGTBRACK:                                                                09250000
                    IF SCAN AND RGTBRACKET THEN                          09251000
                    IF(IF EDITMODE THEN FALSE ELSE SCAN) THEN            09252000
                        IF DELV THEN                                     09253000
                          BEGIN ERR:=DISPLAY(L,U,FPT,FSQ);               09254000
                          DELTOG:=1;                                     09255000
                          END                                            09256000
                        ELSE ERR:=1                                      09257000
                      ELSE ERR:=DISPLAY(L,U,FPT,FSQ)                     09258000
                    ELSE ERR:=2                                          09259000
                    END                                                  09260000
                  ELSE                                                   09261000
                  IF RGTBRACKET THEN                                     09262000
                  IF(IF EDITMODE THEN FALSE ELSE SCAN) THEN              09263000
                      IF DELV THEN                                       09264000
                        BEGIN ERR:=DISPLAY(L,L,FPT,FSQ);                 09265000
                        DELTOG:=1;                                       09266000
                        END                                              09267000
                      ELSE ERR:=3                                        09268000
                    ELSE ERR:=DISPLAY(L,L,FPT,FSQ)                       09269000
                  ELSE ERR:=4                                            09270000
                ELSE ERR:=5                                              09271000
              ELSE                                                       09272000
              IF RGTBRACKET THEN                                         09273000
                BEGIN TADD:=ADDRESS;                                     09274000
                IF SCAN THEN                                             09275000
                  IF IDENT AND ACCUM[0]="6DELETE" THEN                   09276000
                    IF SCAN THEN                                         09277000
                      IF LFTBRACKET THEN                                 09278000
DELOPTION:                                                               09279000
                        IF SCAN AND BOUND(FPT) THEN                      09280000
                          BEGIN U:=ACCUM[0];                             09281000
                          IF SCAN AND RGTBRACKET THEN                    09282000
                            IF SCAN THEN                                 09283000
                              IF DELV THEN                               09284000
                                BEGIN ERR:=DELETE(L,U,FPT,FSQ);          09285000
                                FINISHUP                                 09286000
                                END                                      09287000
                              ELSE ERR:=6                                09288000
                            ELSE ERR:=DELETE(L,U,FPT,FSQ)                09289000
                          ELSE ERR:=7                                    09290000
                          END                                            09291000
                        ELSE ERR:=8                                      09292000
                      ELSE                                               09293000
                      IF DELV THEN                                       09294000
                        BEGIN ERR:=DELETE(L,L,FPT,FSQ);                  09295000
                        FINISHUP                                         09296000
                        END                                              09297000
                      ELSE ERR:=9                                        09298000
                    ELSE ERR:=DELETE(L,L,FPT,FSQ)                        09299000
                  ELSE                                                   09300000
                  IF LFTBRACKET THEN GO TO DELOPTION ELSE                09301000
                    BEGIN CHECKSEQ(SEQ,L,INC);                           09302000
                    CLEANBUFFER(BUFFER,BUFFSIZE,TADD);                   09303000
                    ADDRESS:=ABSADDR(BUFFER); ITEMCOUNT:=0;              09304000
                    IF SCAN THEN GO TO SHORTCUT                          09305000
                    END                                                  09306000
                ELSE ERR:=DELETE(L,L,FPT,FSQ)                            09307000
               END                                                       09308000
              ELSE ERR:=10                                               09309000
```

```
          ELSE ERR:=11                                                    09310000
             END ELSE                                                     09311000
          IF QUADV OR EDITMODE:=(QUOTEQUAD) THEN                          09312000
                BEGIN L:=0; U:=9999.9999; GO TO RGTBRACK                  09313000
          END ELSE                                                        09314000
      IF IOTA THEN                                                        09314200
          IF SCAN AND RGTBRACKET AND FPT NEQ 0 THEN                       09314300
          BEGIN IF SCAN THEN                                              09314310
                IF DELV THEN DELTOG:=1 ELSE ERR:=15;                      09314330
          IF ERR = 0 THEN                                                 09314340
                BEGIN MODE:=RESEQUENCING; CURLINE:=INCREMENT:=1;          09314350
                SETFIELD(GTA,0,8,0);                                      09314400
                GT1:=SEARCHORD(FPT,GTA,GT2,8);                            09314410
                LOWER:=GT2+1; UPPER:=FUNCSIZE-1                           09314420
                END                                                       09314500
          END                                                             09314600
  ELSE ERR:=14                                                            09314700
         ELSE ERR:=12                                                     09315000
      ELSE ERR:=13                                                        09316000
      END                                                                 09317000
    ELSE                                                                  09318000
      IF CURLINE=0 THEN %CHANGING HEADER                                  09318100
      ERR:=26 ELSE                                                        09318110
    IF ERR:=OLDLABCONFLICT(FPT,LINENUMBER(SEQ))=0 THEN                    09319000
      BEGIN                                                               09320000
     IF NOSYNTAX=0 THEN PROCESS(XEQUTE);                                  09321000
      IF ERR:=STOREAWAY(FPT,FSQ,BUFFER,SEQ)=0 THEN SEQ:=SEQ+INC;          09322000
      END;                                                                09323000
      IF ERR NEQ 0 THEN                                                   09324000
  BEGIN FORMWD(2,"5ERROR ");                                              09325000
  NUMBERCON(ERR,ACCUM); ERR:=0;                                           09326000
  EOB:=1;                                                                 09327000
  FORMROW(1,1,ACCUM,2,ACCUM[0],[1:11]);                                   09328000
  END;                                                                    09329000
 END; %OF BLOCK STARTED ON LINE 9238000 ////////////////////////          09330000
  ENDHANDLER:                                                             09330100
      IF BOOLEAN(SUSPENSION) THEN BEGIN                                   09330102
         FILL ACCUM[*] WITH "ABORT SU", "SP. FNS.";                       09330104
         FORMROW(3,0,ACCUM,0,16);  INDENT(0); TERPRINT;                   09330106
         END ELSE                                                         09330108
      IF MODE=0 THEN                                                      09330110
      BEGIN                                                               09330112
      IF BOOLEAN(DELTOG) THEN FINISHUP;                                   09330120
      INDENT(-CURLINE); TERPRINT;                                         09330200
      END;                                                                09330210
                                                                          09331000
  END;                                                                    09332000
  EXPOVR:=FAULTL; INTOVR:=FAULTL; INDEXF:=FAULTL;                         09332100
  FLAG:=FAULTL; ZERO:=FAULTL;                                             09332200
INITIALIZETABLE;                                                          09333000
TRYAGAIN:                                                                 09334000
   IF FALSE THEN %ENTERS WITH A FAULT.                                    09334100
   FAULTL:                                                                09334200
      BEGIN SPOUT(09334300); %SEND A MESSAGE TO SPO                       09334300
                                                                          09334400
         BEGIN CSTATION.APLOGGED:=0; CSTATION.APLHEADING:=0               09334500
         END                                                              09334600
      END;                                                                09334700
   APLMONITOR;                                                            09335000
ENDOFJOB:                                                                 09336000
                                                                          09337000
   FINIS:                                                                 09338000
   WRAPUP;                                                                09339000
                                                                          09340000
END;                                                                      09341000
END;END.        LAST CARD ON OCRDING TAPE                                 99999999


        TOTAL LOGICAL RECORDS=  7273
        END OF JOB.
```

88