

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>

APL /IMAGE
WORDS PER RECORD = 10. WORDS PER BLOCK = 150. TOTAL RECORDS = 7274

CREATION DATE = 71069
DATE OF LAST ACCESS = 72146

REGIN

* THIS APL/R5500 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.

00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000601
00000602
00000603
00000610
00000630
00000700
00000800
00000810
00000900
00001000
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410
00001415
00001416
00001420
00001430
00001440
00001450
00001460
00001465
00001470
00001480
00001490
00001500
00001510
00001520
00001530
00001540

DEFINE VERSIONDATE="1-11-71" #;
* MODIFICATIONS FOR B-5500 TIME-SHARING MCP MADE BY:
* JOSE HERNANDEZ, BURROUGHS CORPORATION.

BOOLEAN BREAKFLAG;
ARRAY G1A(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,INTQVR,INDEX:=INDEXF,FLAG,ZERO;
REAL BIGGEST, NULLV;
INTEGER STACKSIZE,LIBSIZE;
REAL STATUSWORD,CORELOC;
BOOLEAN RETURN;
BOOLEAN MEMRUG,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#,
PAGESPACF=20#,
NEXTTP=[42:6]#,
LASTTP=[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.NEXTTP;
PROCEDURE RPLACD(X,Y); VALUE Y; REAL X,Y; X.NEXTTP:=Y;
BOOLEAN PROCEDURE NULL(X); VALUE X; REAL X; NULL:=X.NEXTTP=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 MARKEDF(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 SETPOINTNAMES;
BEGIN
IF NOT LIBRARIAN(LIBJOB,TIME(-1)) THEN
BEGIN
WRITE(ESTABLISH);
MARKEDF(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 DEBUSED
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."),

```

00001550
00001560
0000157
0000158
00001590
00001600
0000161
0000162
0000163
00001640
00001650
0000166
0000167
00001675
00001680
0000169
0000171
0000173
00001740
00001750
0000176
0000177
00001780
00001790
0000180
0000181
00001820
00001830
0000184
0000185
0000186
00001870
00001880
0000189
0000190
00001910
00001920
0000193
0000194
0000195
00001960
00001970
0000198
0000200
00002010
000020250
0000203
0000204
00002050
0000206
0000207
0000208
0000209
0000210
0000211
0000212
0000213
0000214
0000215
0000216
0000217
0000218
0000219
0000220
0000221
0000222
0000223
0000224
0000225
0000226
0000227
0000228
0000229
0000230
0000231
0000232
0000233
0000234
0000235
0000236
0000237
0000238
0000239
0000240
0000241
0000242
0000243
0000244
0000245
0000246
0000247
0000248
0000249
0000250
0000251
0000252
0000253
0000254
0000255
0000256
0000257
0000258
0000259
0000260
0000261
0000262
0000263
0000264
0000265
0000266
0000267
0000268
0000269
0000270
0000271
0000272
0000273
0000274
0000275
0000276
0000277
0000278
0000279
0000280
0000281
0000282
0000283
0000284
0000285
0000286
0000287
0000288
0000289
0000290
0000291
0000292
0000293
0000294
0000295
0000296
0000297
0000298
0000299
0000300

```

("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(O);
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
NR = "READING THE RECORD," " " " " 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;

```

```

0000297
0000298
0000299
0000300
0000301
0000302
0000303
0000304
0000305
0000306
0000307
0000308
0000309
0000310
0000311
0000312
0000313
0000314
0000315
0000316
0000317
0000318
0000319
0000320
0000321
0000322
0000323
0000324
0000325
0000326
0000327
0000328
0000329
0000330
0000331
0000332
0000333
0000334
0000335
0000336
0000337
0000338
0000339
0000340
0000341
0000342
0000343
0000344
0000345
0000346
0000347
0000348
0000349
0000350
0000351
0000352
0000353
0000354
0000355
0000356
0000357
0000358
0000359
0000360
0000361
0000362
0000363
0000364
0000365
0000366
0000367
0000368
0000369
0000370
0000371
0000372
0000373
0000374

```

```

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)));
NAC 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

```

```

00003750
00003760
00003770
00003780
00003790
00003800
00003810
00003820
00003830
00003840
00003850
00003860
00003870
00003880
00003890
00003900
00003910
00003920
00003930
00003940
00003950
00003960
00003970
00003980
00003990
00004000
00004010
00004020
00004030
00004040
00004050
00004060
00004070
00004080
00004090
00004100
00004110
00004120
00004130
00004140
00004150
00004160
00004170
00004180
00004190
00004200
00004210
00004220
00004230
00004240
00004250
00004260
00004270
00004280
00004290
00004300
00004310
00004320
00004330
00004340
00004350
00004360
00004390
00004400
00004410
00004420
00004430
00004440
00004450
00004460
00004470
00004480
00004490
00004500
00004500
00004600
00004610
00004620
00004630
00004640
00004650
00004660
00004670
00004680

```

```

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(CLESS(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

```

0000469
0000470
0000471
0000472
0000473
0000474
0000475
0000476
0000477
0000478
0000479
0000480
0000481
0000482
0000483
0000484
0000485
0000486
0000487
0000488
0000489
0000490
0000491
0000492
0000493
0000494
0000495
0000496
0000497
0000498
0000499
0000500
0000501
0000502
0000503
0000504
0000505
0000506
0000507
0000508
0000509
0000510
0000511
0000512
0000513
0000514
0000515
0000516
0000517
0000518
0000519
0000520
0000521
0000522
0000523
0000524
0000525
0000526
0000527
0000528
0000529
0000530
0000531
0000532
0000533
0000534
0000535
0000536
0000537
0000538
0000539
0000540
0000541
0000542
0000543
0000544
0000545
0000546
0000547
0000548
0000549
0000550
0000551
0000552
0000553
0000554
0000555
0000556
0000557
0000558
0000559
0000560
0000561

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 DUMP MEMORY (MODE,TYPE,N,M);
IF MODE=8 THEN NPAGES:=NPAGES+N;
MOREPAGES:
BEGIN
OWN BOOLEAN POINTERSET, TYPESET;
INTEGER I, T, NR;
OWN ARRAY RUF[0:MBUFF], TYPES[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
TYPES[I].BF := I; TYPES[T:=INDX[I,0].TF].AF := I;
TYPES[I].BOOL := INDX[I,0].MF;
END;
TYPES[T].BF := I;
END SETTYPES;
REAL PROCEDURE BUFFNUMBER(I); VALUE I; INTEGER I;
BEGIN INTEGER K,L,M;
LABEL D;
DEFINE B=BUF#;
IFC 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;
RUF[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 -- INITIALIZE 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);

```

```

00005620
00005630
0000567
0000568
00005690
00005693
00005698
0000570
00005704
0000570
0000570
0000571
00005712
00005714
0000571
0000571
0000572U
00005730
0000574
0000575
0000576
00005770
00005780
0000579
0000580
0000581U
00005820
0000583
0000584
0000585
00005860
00005870
0000588
0000589
0000590U
00005910
0000592
0000593
0000594U
00005950
00005960
0000597
0000598
00005990
00006000
0000601
0000602
0000603
00006040
00006050
0000606
0000607
00006080
00006090
0000610
0000611
0000612U
00006130
0000614
0000615
0000616
00006170
00006180
0000619
0000620
0000621U
00006220
0000623
0000624
0000625
00006260
00006270
0000628
0000629
0000630U
00006310
0000632
0000633
0000634
00006350
00006360

```



```

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 ----- RESERVED -----
  ;
  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;

```

```

00006370
00006380
00006390
00006400
00006410
00006550
00006560
00006565
00006570
00006580
00006590
00006600
00006610
00006620
00006630
00006640
00006650
00006660
00006670
00006680
00006690
00006700
00006710
00006720
00006730
00006740
00006750
00006760
00006770
00006780
00006790
00006800
00006810
00006820
00006830
00006840
00006850
00006860
00006865
00006870
00006880
00006890
00006900
00006910
00006920
00006930
00006940
00006950
00006960
00006965
00006970
00006980
00006985
00006990
00006995
00007000
00007010
00007020
00007030
00007040
00007050
00007060
00007070
00007080
00007090
00007100
00007110
00007120
00007130
00007140
00007150
00007160
00007170
00007180
00007190
00007200
00007210
00007220
00007230
00007240
00007250
00007260

```



```

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+1; 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+1),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;

```

```

00007860
00007870
00007880
00007890
00007900
00007910
00007920
00007922
00007925
00007924
00007927
00007930
00007940
00007945
00007947
00007950
00007960
00007970
00007980
00007985
00007990
00008000
00008010
00008020
00008030
00008040
00008050
00008060
00008070
00008080
00008090
00008100
00008110
00008120
00008130
00008140
00008150
00008160
00008170
00008180
00008190
00008200
00008210
00008220
00008230
00008240
00008250
00008260
00008262
00008270
00008280
00008290
00008300
00008310
00008320
00008330
00008340
00008350
00008360
00008370
00008380
00008390
00008400
00008410
00008420
00008430
00008440
00008450
00008460
00008470
00008480
00008490
00008500
00008510
00008520
00008530
00008540
00008550
00008560
00008570
00008580
00008590
00008600

```

```

ARRAY R[0:PAGE SIZE-1];
COMMENT PAGE SIZE -1 SHOULD BE COMPUTED TO THE EXACT
NUMBER OF WORDS TO MOVE -- IT WOULD SPEED THINGS UP;
INTEGER L;
I:=INDX[CURPAGE,0]; COMMENT I,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,PAGE SIZE);
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);
TYPES[0].BF:=FREEPAGE(INDX,TYPES[0].BF,CURPAGE,CURPAGE);
UPDATE(TYPES,1,TYPE-1,1); TYPES[TYPE].AF:=T.AF+1;
COMMENT
IF MEMBUG.[6:1] THEN DUMPTYPES(MODE,TYPES,NTYPES);
END

```

```

END;
%----- MODE=7 ----- SEARCH FOR A RECORD FROM THE FILE ---

```

```

COMMENT IF N GTR 3 THEN MESSAGE(14) ELSE
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:=TYPES[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 R[0:RECSIZE-1];
U:=TYPES[TYPE].BF; L:=TYPES[TYPE].AF;
WHILE U-L GTR 1 DO
IF LESS(A,0,INDX[F:=(U+L) DIV 2,0],B,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 RELATIVE POSITION OF THE CLOSEST RECORD IN THIS PAGE.
N=1	"	DO NOT PLACE IN FILE. RETURN ABSOLUTE SUBSCRIPT OF CLOSEST 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

```

```

00008610
00008620
00008630
00008640
00008650
00008660
00008670
00008680
00008690
00008700
00008710
00008720
00008730
00008740
00008750
00008760
00008770
00008780
00008790
00008800
00008810
00008820
00008830
00008840
00008850
00008860
00008870
00008880
00008890
00008900
00008910
00008920
00008930
00008940
00008950
00008960
00008970
00008980
00008990
00009000
00009010
00009020
00009030
00009040
00009050
00009060
00009070
00009080
00009090
00009100
00009110
00009120
00009130
00009140
00009150
00009160
00009170
00009180
00009190
00009200
00009210
00009220
00009230
00009240
00009250
00009260
00009270
00009280
00009290
00009300
00009310
00009320
00009330
00009340
00009350
00009360
00009370
00009380
00009390
00009400

```

```

IF LESS(B,0,A,0,M) THEN M:=2 ELSE
M:=0;
T:=0; IF BOOLEAN(N) THEN
FOR I:=TYPST[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:PAGE SIZE-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],PAGE SIZE,A[*]); INDX[I,P],PAGEF:=I END;
MARKEOF(SKIP,RECSIZE,NEWDISK(0));
WRITE(NEWDISK[I]);
TYPST[0].BF:=FREEPAGE(INDX,TYPST[0].BF,T,NPAGES);
UPDATE(TYPST,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:PAGE SIZE-1];
COMMENT
MONITOR PRINT(Q,W,N,I,J,K,T1,T2,T3,M,A,B);
IF I:=TYPST[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:=TYPST[Q,T1]).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),
PAGE SIZE,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,PAGE SIZE,POINTERS(CURBUFF:=BUFFNUMBER
(I))(0)); SORT(INDX,0,NPAGES,RECSIZE*B);
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;

```

```

00009410
00009420
00009430
00009440
00009450
00009460
00009470
00009480
00009490
00009500
00009510
00009520
00009530
00009540
00009550
00009560
00009570
00009580
00009590
00009600
00009610
00009620
00009630
00009640
00009650
00009660
00009670
00009680
00009690
00009700
00009710
00009720
00009730
00009740
00009750
00009760
00009770
00009780
00009790
00009800
00009810
00009820
00009830
00009840
00009850
00009860
00009870
00009880
00009890
00009900
00009910
00009920
00009930
00009940
00009950
00009960
00009970
00009980
00009990
00010000
00010010
00010020
00010030
00010040
00010050
00010060
00010070
00010080
00010090
00010100
00010110
00010120
00010130
00010140
00010150
00010160
00010170
00010180
00010190
00010200
00010210
00010220
00010230

```



```

IF I GIR NPAGES THEN N:=REAL(WRITEBUFFER);
END OF FILE UPDATE;
*----- MODE=10 ----- EMERGENCY FILE MAINTENANCE -----
DO MEMORY(9,TYPE,A,N,M) UNTIL N NEO 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:=TYPST[TYPE]).AF=T.BF THEN TYPST[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(MPBUF-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
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:=TYPST[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;
TYPST[0].BF:=FREEPAGE(INDX,TYPST[0].BF,T.AF,J);
UPDATE(TYPST,1,TYPE-1,J-T.AF+1);
TYPST[TYPE].BF:=T.AF; TYPST[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;

```

```

00010240
00010250
00010260
00010270
00010280
00010290
00010300
00010310
00010320
00010330
00010340
00010350
00010360
00010370
00010380
00010390
00010400
00010410
00010420
00010430
00010440
00010450
00010460
00010470
00010480
00010490
00010500
00010510
00010520
00010530
00010540
00010550
00010560
00010570
00010580
00010590
00010600
00010610
00010620
00010630
00010640
00010650
00010660
00010670
00010680
00010690
00010700
00010710
00010720
00010730
00010740
00010750
00010760
00010770
00010780
00010790
00010800
00010810
00010820
00010830
00010840
00010850
00010860
00010870
00010880
00010890
00010900
00010910
00010920
00010930
00010940
00010990
00011000
00011010
00011020
00011030
00011040
00011050
00011060
00011070
00011080
00011090
00011100

```

```

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#,
ACCOUNT=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]#,

```

```

00011111C
00011112C
00011113C
00011133C
00011134C
00011135C
00011136C
00011137C
00011138C
00011139C
00011140C
00011141C
00011142C
00011143C
00011144C
00011145C
00011146C
00011151C
00011156C
00011157C
00011163C
00011165C
00011166C
00011167C
00011173C
00011180C
00011181C
00011182C
00011183C
00011190C
00011192C
00011193C
00011194C
00011195C
00011196C
00011197C
00011198C
00011199C
00012000C
00012010C
00012070C
00012100C
00012110C
00012120C
00012130C
00012140C
00012420C
00012430C
00012440C
00012450C
00012460C
00012470C
00012570C
00012580C
00012590C
00012600C
00012610C
00012620C
00012630C
00012640C
00013000C
00013010C
00013020C
00013022C
00013030C
00013040C
00013042C
00013050C
00013060C
00013070C
00013080C
00013090C
00013092C
00013100C
00013110C
00013112C
00013114C
00013116C
00013118C
00013120C
00013130C
00013140C
00013150C

```

```

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=[18:81]#,
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]#,
DDPNVW=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#, % PRG 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)
OUTDF=1#,
NAME DNULLV=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#,
LINERUFFER=LINERUFFER#,
LINERUFFER=OUTRUFF[*]1#,
APPENDTORUFFER=APPENDTOBUFF#,

```

```

00013160
00013170
00013180
00013190
00013200
00013210
00013220
00013230
00013240
00013250
00013260
00013270
00013280
00014000
00015000
00016000
00017000
00017100
00017110
00017120
00017130
00017140
00017150
00017160
00017170
00017200
00017210
00017220
00017230
00017240
00017250
00017260
00017270
00017280
00017290
00018000
00018100
00018200
00020000
00020100
00021000

```

```

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]#,
TABSIZ = 43#,
LOGINCODES=1#,
LOGINPHRASE=2#,
LIBRARY=1#,
WORKSPACEUNIT=2#,
RTPAREN=9#,
MASTERMODE=USERMASK.[1:1]#,
EDITOG=USERMASK.[2:1]#,
PDLBUG=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]#,
NUMBERARGS=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#,
DROPTIT = 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#,
RGT PAREN = 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#,
BOOL AND = SPECIAL AND ACCUM[0] = 21#,
BOOLOR = SPECIAL AND ACCUM[0] = 22#,
BOOLOR NOT = SPECIAL AND ACCUM[0] = 23#,
LESS THAN = SPECIAL AND ACCUM[0] = 24#,
LESSEQ = SPECIAL AND ACCUM[0] = 25#,
EQUAL = SPECIAL AND ACCUM[0] = 26#,
GREATER = SPECIAL AND ACCUM[0] = 27#,
GREATER NOT = 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#,

```

```

00022000
00023000
00024000
00025000
00026000
00027000
00028000
00029000
00029100
00030000
00030100
00030200
00030210
00030220
00030300
00030400
00030401
00030402
00030403
00030404
00030405
00030406
00030407
00030408
00030409
00030410
00030411
00030412
00030413
00030414
00030415
00030416
00030417
00030418
00030419
00030420
00030421
00030500
00030510
00030600
00030610
00030620
00030621
00030622
00030623
00030624
00030625
00030626
00030627
00030628
00030629
00030630
00030631
00030632
00030633
00030634
00030635
00030636
00030637
00030638
00030639
00030640
00030641
00030642
00030643
00030644
00030645
00030646
00030647
00030648
00030649
00030650
00030651
00030652
00030653
00030654
00030655
00030656
00030657
00030658
00030659
00030660
00030661
00030662
00030663
00030664
00030665
00030666
00030667
00030668
00030669
00030670
00030671
00030672
00030673
00030674
00030675
00030676
00030677
00030678
00030679
00030680
00030681
00030682
00030683
00030684
00030685
00030686
00030687
00030688
00030689
00030690
00030691
00030692
00030693
00030694
00030695
00030696
00030697
00030698
00030699
00030700
00030701
00030702
00030703
00030704
00030705
00030706
00030707
00030708
00030709
00030710
00030711
00030712
00030713
00030714
00030715
00030716
00030717
00030718
00030719
00030720
00030721
00030722
00030723
00030724
00030725
00030726
00030727
00030728
00030729
00030730
00030731
00030732
00030733
00030734
00030735
00030736
00030737
00030738
00030739
00030740
00030741
00030742
00030743
00030744
00030745
00030746
00030747
00030748
00030749
00030750
00030751
00030752
00030753
00030754
00030755
00030756
00030757
00030758
00030759
00030760
00030761
00030762
00030763
00030764
00030765
00030766
00030767
00030768
00030769
00030770
00030771
00030772
00030773
00030774
00030775
00030776
00030777
00030778
00030779
00030780
00030781
00030782
00030783
00030784
00030785
00030786
00030787
00030788
00030789
00030790
00030791
00030792
00030793
00030794
00030795
00030796
00030797
00030798
00030799
00030800
00030801
00030802
00030803
00030804
00030805
00030806
00030807
00030808
00030809
00030810
00030811
00030812
00030813
00030814
00030815
00030816
00030817
00030818
00030819
00030820
00030821
00030822
00030823
00030824
00030825
00030826
00030827
00030828
00030829
00030830
00030831
00030832
00030833
00030834
00030835
00030836
00030837
00030838
00030839
00030840
00030841
00030842
00030843
00030844
00030845
00030846
00030847
00030848
00030849
00030850
00030851
00030852
00030853
00030854
00030855
00030856
00030857
00030858
00030859
00030860
00030861
00030862
00030863
00030864
00030865
00030866
00030867
00030868
00030869
00030870
00030871
00030872
00030873
00030874
00030875
00030876
00030877
00030878
00030879
00030880
00030881
00030882
00030883
00030884
00030885
00030886
00030887
00030888
00030889
00030890
00030891
00030892
00030893
00030894
00030895
00030896
00030897
00030898
00030899
00030900
00030901
00030902
00030903
00030904
00030905
00030906
00030907
00030908
00030909
00030910
00030911
00030912
00030913
00030914
00030915
00030916
00030917
00030918
00030919
00030920
00030921
00030922
00030923
00030924
00030925
00030926
00030927
00030928
00030929
00030930
00030931
00030932
00030933
00030934
00030935
00030936
00030937
00030938
00030939
00030940
00030941
00030942
00030943
00030944
00030945
00030946
00030947
00030948
00030949
00030950

```

RHD	== SPECIAL AND ACCUM[0] = 34#	00030950
IOTA	== SPECIAL AND ACCUM[0] = 35#	00030960
TRACE	== SPECIAL AND ACCUM[0] = 36#	00030970
PHI	== SPECIAL AND ACCUM[0] = 37#	00030980
EXPAND	== SPECIAL AND ACCUM[0] = 38#	00030981
BASVAL	== SPECIAL AND ACCUM[0] = 39#	00030982
EXCLAMATION	== SPECIAL AND ACCUM[0] = 40#	00030983
MINUSLASH	== SPECIAL AND ACCUM[0] = 41#	00030984
QUESTION	== SPECIAL AND ACCUM[0] = 42#	00030985
OSLASH	== SPECIAL AND ACCUM[0] = 43#	00030986
TAU	== SPECIAL AND ACCUM[0] = 44#	00030987
CIRCLE	== SPECIAL AND ACCUM[0] = 45#	00030988
LOCKIT	= IDENT AND ACCUM[0] = "4 LOCK "#	00030989
COLON	= SPECIAL AND ACCUM[0] = 47#	00030990
QUADLFTARROW	= 51#	00030992
REDUCT	= 52#	00030993
ROTATE	= 53#	00030994
SCANV	= 57#	00030995
LINEBUFFSIZE	= 17#	00031000
MAXPOLISH	= 100#	00031002
MESSIZE	= 10#	00031004
MAXCONSTANT	= 30#	00031005
MAXMEMACCESSSES	= 3584#	00031006
%MAXSPROWS	x SPRSIZE	00031007
MAXSYMBOL	= 30#	00031008
MAXSPROWS	= 28#	00031009
TYPEFIELD	= [3:3]#	00031010
OPTYPE	= [1:2]#	00031011
LDCFIELD	= BACKP#	00031012
ADDRFIELD	= SPF#	00031013
SYMTYPE	= [3:3]#	00031014
OPERAND	= 5#	00031016
CONSTANT	= 2#	00031018
OPERATOR	= 3#	00031019
LOCALVAR	= 4#	00031020
SYMTABSIZE	= 1#	00031022
LFTPAREN	= 8#	00031024
RGTAREN	= 9#	00031026
LFTBRACKETV	= 11#	00031028
RGTBRACKETV	= 12#	00031030
SEMICOLONV	= 15#	00031032
QUAD	= 10#	00031033
QUAD	= 14#	00031034
LFTARROWV	= 4#	00031035
SDRTUPV	= 55#	00031036
SDRTDNV	= 56#	00031040
ALPHALABEL	= 1#	00031050
NUMERICLABEL	= 2#	00031060
NEXTLINE	= 0#	00031062
ERRORCOND	= 3#	00031070
PRESENCE	= [2:1]#	00031080
CHANGE	= [1:1]#	00031090
XEQ	= 1#	00031092
CLEARCORE	= 2#	00031094
WRITECORE	= 3#	00031096
XXXX		00031098
XXXX		00031098
XEQUTE	= 1#	00031100
SLICE	= 120#	00031102
%TIME SLICE	IN 60THS OF A SECOND	00031104
ALLOC	= 2#	00031106
WRITEBACK	= 3#	00031108
LOOKATSTACK	= 5#	00031110
LEN	= [1:23]#	00032000
NEXT	= [24:24]#	00032002
LOC	= L.[30:11], L.[41:7]#	00032004
NOC	= N.[30:11], N.[41:7]#	00032008
MDC	= M.[30:11], M.[41:7]#	00032010
SPRSIZE	= 128#	00032015
% SP ROW SIZE		00032020
NILADIC	= 0#	00032030
MONADIC	= 1#	00032040
DYADIC	= 2#	00032050
TRIADIC	= 3#	00032100
DEPTHERROR	= 1#	00032110
DOMAINERROR	= 2#	00032120
INDEXERROR	= 4#	00032130
LABELERROR	= 5#	00032140
LENGTHERROR	= 6#	00032150
NONCEERROR	= 7#	00032160
RANKERROR	= 8#	00032170
SYNTAXERROR	= 9#	00032180
SYSTEMERROR	= 10#	00032190
VALUEERROR	= 11#	00032200
SPERROR	= 12#	00032200
KITEERROR	= 13#	00032201


```

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="0" 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
    ;

```

```

00059100
00059800
00059810
00059820
00059830
00059840
00059900
00060000
00061000
00062000
00063000
00064000
00065000
00066000
00067000
00068000
00069000
00070000
00071000
00072000
00072100
00072200
00073000
00074000
00075000
00076000
00077000
00078000
00079000
00080000
00081000
00082000
00083000
00084000
00085000
00086000
00087000
00088000
00089000
00090000
00091000
00092000
00093000
00094000
00095000
00095100
00095110
00095120
00095130
00095140
00095150
00095160
00095170
00095180
00095190
00095200
00095210
00095220
00095230
00095240
00095250
00095260
00095270
00095280
00095290
00095300
00095310
00095330
00095340
00095350
00095360
00095370
00095380
00095390
00095400
00095410
00095420
00095430
00095440
00095450
00096000
00096100
00096200

```

```

PROCEDURE INDEXT(R); VALUE R; REAL R; FORWARD; 00096300
PROCEDURE TERPRINT; FORWARD; 00096400
PROCEDURE PROCFS(MODE); VALUE MODE; INTEGER MODE; FORWARD; 00096500
REAL STREAM PROCEDURE ABSADDR(A); 00097000
BEGIN SI:=A; ABSADDR:=SI 00098000
END; 00099000
BOOLEAN PROCEDURE LIBRARIAN(MFID,FID); VALUE MFID,FID; 00099100
REAL MFID,FID; 00099110
BEGIN 00099120
REAL ARRAY A(0:6); FILE DF DISK(1,1); 00099125
REAL T; 00099130
COMMENT LIBRARIAN IS TRUE IF MFID/FID IS PRESENT ON DISK; 00099137
FILL DF WITH MFID,FID; 00099140
SEARCH(DF,A[*]); 00099145
LIBRARIAN:= 00099150
A(0)=-1; 00099160
END; 00099170
FILE SPO 11(1,3); 00099300
PROCEDURE SPOUT(K); VALUE K; INTEGER K; 00099310
BEGIN FORMAT ERRF("APL ERROR:",I8,A1); 00099320
WRITE(SPO,ERRF,K,31); 00099330
END; 00099340
PROCEDURE INITIALIZETABLE; 00100000
BEGIN DEFINE STARTSEGMENT= #; 00101000
INTEGER I; 00101005
LADDRESS:= 00101010
ABSOLUTEADDRESS:=ABSADDR(BUFFER); 00101100
BIGGEST := REAL(NOT FALSE) & 0(1:46:2); 00101200
NULLV := 0 & 3(1:46:2); 00101300
STATUSWORD+REAL(BOOLEAN(STATUSWORD) OR BOOLEAN(1)); 00101400
JOBNUM+TIME(-1); 00101410
STATION+0&1((CLOGGED)&STATUSWORD[STU]); 00101420
FILL JIGGLE[*] WITH OCT5757575757575737;%CARRIAGE RETURNS LEFT ARROW 00101430
FILL IDTABLE[*] WITH 00102000
"1+481-49", "1&501&07", "1.171@19", "1#411(08", 00103000
"1)091/06", "3XEQ623L", "0G541;15", OCT0333777601040177, 00103100
%LAST IN ABOVE LINE IS REALLY 3(1141" 00103200
"202:=042", "11101(11", "11123AND", "2120R223", 00103300
"NOT233LS", "S243LEQ2", "53GEQ273", "GTR283NE", "Q292=:05", 00103350
"2G0051=2", "63MAX304", "CEIL303F", "LR313MIN", 00103400
"314RES03", "23ABS323", "RHO341*1", "84INTA35", 00103500
"1x384RND", "M425TRAN", "S431$133", "PHI374FA", 00103600
"RT404COM", "B406CIRC", "LE456SUR", "TUP556SO", 00103700
"RTDN561:", "474NAND5", "83NDR594", "TAKE604D", 00103800
"ROP613RE", "P446BASV", "AL393EPS", "331,1600"; 00103900
COMMENT IDTABLE IS TABLE OF RESERVED WORDS AND SPECIAL SYMBOLS, 00103910
FORMAT IS NUMBER OF CHARACTERS IN SYMBOL, FOLLOWED BY SYMBOL 00103913
ITSELF, FOLLOWED BY A TWO-DIGIT DECIMAL CODE WHICH APL USES 00103916
FOR THE RESERVED WORD--LIKE IN THE EXECUTION CASE STATEMENT AND 00103919
IN SYNTAX CHECKING. FOR SCAN TO WORK, THE TWO-DIGIT CODE MUST 00103922
BE GREATER THAN 3 AND IDTABLE MUST HAVE AT LEAST ONE "0" AT THE 00103925
END TO MARK THE END. TABSIZE IS THE DEFINE (LINE 30000) GIVING 00103928
THE SIZE OF IDTABLE; 00103931
IF STACKSIZE=0 THEN STACKSIZE:=100 ELSE 00103940
IF STACKSIZE GTR 1022 THEN STACKSIZE:=1022; 00103950
BUFFSIZE:=MAXBUFFSIZE; 00104000
LINETOG := TRUE; %USUALLY GO TO NEXT LINE WHEN WRITING OUT 00104010
INITBUFF(OUTBUFF, 10); 00104100
INITBUFF(BUFFER,BUFFSIZE); 00104500
NROWS:=-1; 00105000
NAME(LIRJOR,TIME(-1)); 00105100
FILL MESSTAB[*] WITH 00105200
"4SAVE ", 00105210
"4LOAD ", 00105220
"5CLEAR ", 00105230
"4COPY ", 00105240
"4VARS ", 00105250
"3FNS ", 00105260
"6LOGGED", 00105270
"3MSG ", 00105280
"5WIDTH ", 00105290
"3OPR ", 00105300
"6DIGITS", 00105310
"3OFF ", 00105320
"6ORIGIN", 00105322
"4SEED ", 00105324
"4FUZZ ", 00105326
"3SYN ", 00105328
"5NOSYN ", 00105330
"5STORE ", 00105332
"5ABORT ", 00105334
"2SI ", 00105335

```

```

"3SIV ";
"5ERASE ";
x-----MASTERMODE BELOW HERE...(SEE USERTOP)-----
"6ASSIGN";
"6DELETE";
"4LIST ";
"5DEBUG ";
"5FILES ";

```

```

IF LIBSIZE=-1 THEN
BEGIN LIBSIZE+1;GTA[0]←" ";STOREDRD(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(LOGINCODS)=0) THEN
END;
END;

```

```

FILL CORRESPONDENCE[*] WITH
OCT111111111111110311,
OCT1111111111111111,
OCT1104111121221113,
OCT2014151617100706,
OCT1111111111111112,
OCT1111111111111100,
OCT020111111251111,
OCT232411111111111;

```

```

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;

```

```

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,10000000,0,0,x,CONV(ADDR,8),
0,+,:=,THI,TLO);
T:=THI
END ELSE T:=T×10000000 + 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);

```

00105360
00105370
00105380
00105390
00105400
00105410
00105420
00105440
00106000
00106090
00106091
00106093
00106094
00106095
00106096
00106099
00106100
00106102
00106104
00106500
00106510
00106520
00106530
00106540
00106550
00106560
00106570
00106571
00106573
00106575
00106577
00106579
00106581
00106583
00106584
00106586
00106588
00107000
00108000
00108500
00109000
00109500
00110000
00110500
00111000
00111500
00112000
00112500
00113000
00113500
00114000
00114500
00115000
00115500
00116000
00116500
00117000
00117500
00118000
00118500
00119000
00119500
00120000
00120500
00121000
00121500
00122000
00122500
00123000
00123500
00124000
00124500
00125000
00125500
00126000
00126500
00127000
00127500
00128000
00128500
00129000
00129500
00130000

```

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;
BK: IF FALSE THEN
IF CURRENTMODE=XEQMODE THEN BREAKFLAG:=TRUE;
END OF TERPRINT;
PROCEDURE FORMWD(CC,WD); VALUE CC,WD; REAL WD; INTEGER CC;
BEGIN
INTEGER T,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.

```

```

00130500
00131000
00131500
00132000
00132500
00133000
00133500
00134000
00134500
00135000
00135500
00136000
00136500
00137000
00137500
00138000
00138500
00139000
00139500
00140000
00140500
00141000
00141500
00142000
00142500
00143000
00143500
00144000
00144500
00145000
00145500
00146000
00146500
00147000
00147500
00148000
00148500
00149000
00149500
00150000
00150500
00151000
00151500
00152000
00152500
00220000
00221000
00222000
002223000
002224000
002225000
002226000
002227000
002228000
002229000
00230000
00231000
00231030
00231040
00232000
00232100
00232200
00232300
00232400
00232500
00232600
00232700
00240000
00241000
00242000
00242500
00243000
00243500
00243600
00243610
00244000
00244100
00245000
00253000
00254000
00255000
00255090
00256000

```



```

CC=1 STAY ON THIS LINE BUT TERMINATE PRINT.
CC=2 SKIP TO NEXT LINE - MORE TO COME.
CC=3 SKIP TO NEXT LINE - TERMINATE PRINT.;
REAL STREAM PROCEDURE OCTAL(I); VALUE I;
  BEGIN SI:=LOC I; DI:=LOC OCTAL; DS:=8OCT
  END;
IF L:=LINESIZE LEQ 9 OR L GTR 72 THEN L:=72; K:=2;
IF CC GTR 1 AND CHRCOUNT GTR OTHER TERPRINT;
IF CHRCOUNT+(I:=OCTAL(WD.(1:11))) GTR L THEN
  BEGIN APPENDTOBUFF(LINEBUFFER,CHRCOUNT,
    0,WD,2,K:=L-CHRCOUNT);
  CHRCOUNT:=L; TERPRINT;
  I:=I-K;
  END;
APPENDTOBUFF(LINEBUFFER,CHRCOUNT,0,WD,K,I);
CHRCOUNT:=CHRCOUNT+I;
IF BOOLEAN(CC) THEN
  IF CC=-1 THEN BEGIN LINETOQ:=FALSE;
    TERPRINT; LINETOQ:=TRUE
  END ELSE TERPRINT;
END;
BOOLEAN PROCEDURE FUNCTIONHEADER(SPECS,HADDR);
ARRAY SPECS[0]; REAL HADDR; FORWARD;

REAL PROCEDURE LINENUMBER(R); VALUE R; REAL R;
COMMENT STARTS ON 8030000;
FORWARD;

PROCEDURE INDENT(R); VALUE R; REAL R;
BEGIN
  INTEGER STREAM PROCEDURE FORM(A,I,K); VALUE K,I;
  BEGIN
    LOCAL T1,T2;
    LABEL SHORT,L,M,FINIS;
    TALLY:=K; FORM:=TALLY;
    SI:=LOC I; DI:=LOC T1; IF RSC=DC THEN
      BEGIN DI:=A; K(DS:=LIT" "); GO FINIS
    END;
    SI:=LOC I; DI:=A; TALLY:=3; DS:=LIT"[";
    IF SC GTR "0" THEN IF SC LSS "0" THEN
      3(TALLY:=TALLY+1; IF TOGGLE THEN DS:=CHR ELSE
        IF SC NEQ "0" THEN DS:=CHR ELSE
          BEGIN TALLY:=TALLY+63; SI:=SI+1
          END );
    DS:=CHR; T1:=TALLY; TALLY:=4; SI:=SI+3;
    4(IF SC NEQ "0" THEN JUMP OUT TO M;
      TALLY:=TALLY+63; SI:=SI-1); GO TO L;
    M;
    T2:=TALLY; SI:=LOC I; SI:=SI+4; DS:=LIT". "; DS:=T2 CHR;
    TALLY:=T1; TALLY:=TALLY+T2; TALLY:=TALLY+1; T1:=TALLY;
    L;
    DS:=LIT"]"; TALLY:=K;
    T1(TALLY:=TALLY+63; T2:=TALLY; SI:=LOC T2; SI:=SI+7;
      IF SC="0" THEN JUMP OUT TO SHORT);
    T2(DS:=LIT" "); GO FINIS;
    SHORT;
    TALLY:=T1; TALLY:=TALLY+1; FORM:=TALLY; DS:=LIT" ";
    FINIS;
    DS:=RESET; DS:=5SET;
  END;
  IF R LSS 0 THEN R:=LINENUMBER(-R) ELSE R:=ABS(R); % -0
  CHRCOUNT:=FORM(LINEBUFF,R,MARGINSIZE)+1
END;
INTEGER PROCEDURE HEADER(ADDR1,ADDR2,BUF); VALUE ADDR1,ADDR2;
INTEGER ADDR1, ADDR2; ARRAY BUF[0];
BEGIN
  INTEGER STREAM PROCEDURE HEADRR(ADDR1,ADDR2,BUF); VALUE ADDR1,
  ADDR2;
  BEGIN
    LOCAL C,T,TDI;
    LOCAL QM,AR;
    LABEL L,ENDSCAN,M,N;
    DI:=LOC QM; DS:=2RESET; DS:=2SET;
    DI:=LOC AR; DS:=RESET; DS:=5SET;
    DI:=BUF;
    SI:=ADDR1;

```

```

00257000
00258000
00259000
00260000
00261000
00262000
00263000
00264000
00265000
00266000
00267000
00268000
00269000
00270000
00271000
00272000
00273000
00274000
00274900
00275000
00276000
00276010
00276020
00276030
00277000
00277500
00277600
00278000
00279000
00280000
00280100
00280110
00280120
00280130
00281000
00281100
00281200
00281300
00281400
00281500
00281600
00281700
00281800
00281900
00282000
00282100
00282200
00282300
00282400
00282500
00282600
00282700
00282800
00282900
00283000
00283100
00283200
00283300
00283400
00283500
00283600
00283700
00283800
00283900
00284000
00284100
00285000
00286000
00286100
00287000
00287010
00287020
00287030
00287100
00287110
00287120
00287130
00287132
00287140
00287142
00287144
00287180
00287200

```

```

L: T:=SI; TDI:=DI;
DI:=LOC QM; IF SC=DC THEN GO TO ENDSCAN;
SI:=LOC AR; SI:=SI-1; IF SC=DC THEN GO TO ENDSCAN;
SI:=LOC T; DI:=LOC ADDR2;
IF 8SC=DC THEN COMMENT END OF SCAN;
GO TO ENDSCAN;
SI:=T; DI:=TDI; DS:=CHR;
GO TO L;
ENDSCAN;
SI:=TDI;
M: SI:=SI-1;
IF SC=" " THEN GO TO M;
SI:=SI+1;
ADDR2:=SI;
SI:=BUF;
N: T:=SI; DI:=LOC ADDR2;
SI:=LOC T;
IF 8SC NEQ DC THEN
BEGIN
TALLY:=TALLY+1; TDI:=TALLY;
SI:=LOC TDI; SI:=SI+7;
IF SC="0" THEN
BEGIN TALLY:=C; TALLY:=TALLY+1; C:=TALLY;
TALLY:=0;
END;
SI:=T; SI:=SI+1; GO TO N;
END;
HEADRR:=TALLY; SI:=LOC C; DI:=LOC HEADRR; SI:=SI+1; DS:=6 CHR;
END;
HEADER:=HEADRR(ADDR1,ADDR2,BUF);
END OF PHONY HEADER;
PROCEDURE STARTSCAN;
BEGIN

```

```

LADDRESS:=
ADDRESS:=ABSOLUTEADDRESS;
BEGIN TERPRINT;
END;
READ(TXIN(STOP),29,BUFFER[*]);
BUFFER[30]:=0&31[1:43:5];
ITEMCOUNT:=0;
EOR:=0;
END;
PROCEDURE FORMROW(CC,BL,A,S,N); VALUE CC,BL,S,N; INTEGER CC,BL,
S,N; ARRAY A[0];
COMMENT: CC--SAME CODE AS IN FORMWD, LINE 253000
BL--#BLANKS TO PUT IN FRONT OF IT
A--ARRAY WHERE THE STUFF TO PUT ON LINE IS STORED
S--#CHARACTERS TO SKIP AT START OF A
N--#CHARACTERS TO TAKE FROM A TO PUT ON OUTPUT LINE;
BEGIN INTEGER K;
INTEGER T;
IF CC GTR 1 AND CHRCOUNT GTR 0 THEN TERPRINT;
IF K:=LINESIZE LEQ 9 OR K GTR 72 THEN K:=72;
WHILE CHRCOUNT+N+BL GTR K DO
BEGIN
APPENDTOBUFFER(LINEBUFFER,CHRCOUNT,BL,A,S,T:=K-CHRCOUNT-BL);
CHRCOUNT:=K; TERPRINT;
S:=S+T; N:=N-T;
BL:=0;
END;
APPENDTOBUFFER(LINEBUFFER,CHRCOUNT,BL,A,S,N);
CHRCOUNT:=CHRCOUNT+N+BL;
IF BOOLEAN(CC) THEN
IF CC=-1 THEN BEGIN LINETOGL:=FALSE;
TERPRINT; LINETOGL:=TRUE;
END ELSE TERPRINT;

```

```

END;
PROCEDURE NUMBERCON(R,A); VALUE R; REAL R; ARRAY A[0];
BEGIN FORMAT F(F24,*), G(E24,*);
REAL S; DEFINE MAXIM = 10@9#;
STREAM PROCEDURE ADJUST(A,B);
BEGIN LOCAL T,FRAC,MANT,T1,TSI,TDI;
DI:=LOC T; DI:=DI+1; T1:=DI;
SI:=R; DI:=A; DI:=DI+2;
24(IF SC=" " THEN SI:=SI+1 ELSE
BEGIN TSI:=SI; SI:=LOC T;
IF SC="1" THEN; SI:=TSI;
IF TOGGLE THEN

```

```

00287210
00287212
00287214
00287220
00287230
00287240
00287250
00287260
00287300
00287310
00287320
00287330
00287332
00287340
00287350
00287360
00287370
00287380
00287390
00287400
00287410
00287420
00287430
00287440
00287450
00287460
00287470
00287480
00287490
00287492
00287494
00299000
00300000
00300100
00300600
00300700
00301000
00302000
00304000
00305000
00306000
00307000
00312000
00313000
00314000
00315000
00316000
00316010
00316020
00316030
00316040
00316050
00317000
00317100
00318000
00319000
00320000
00321000
00322000
00323000
00324000
00325000
00326000
00327000
00327900
00328000
00329000
00329010
00329020
00329030
00330000
00331000
00332000
00332010
00333000
00334000
00335000
00336000
00337000
00338000
00339000
00340000
00341000

```

```

IF SC NEQ "0" THEN
  IF SC="@" THEN BEGIN
    TSI:=SI; DI:=T1; DS:=LIT"1"; JUMP OUT;
    END ELSE FRAC:=TALLY
    ELSE TALLY := TALLY+0
  ELSE
    IF SC="." THEN
      BEGIN MANT:=TALLY; TDI:=DI; DI:=LOC T; DS:=
        LIT"1"; TALLY:=0; DI:=TDI;
        END;
      TALLY:=TALLY+1; DS:=CHR
    END);
  SI:=LOC MANT; SI:=SI+7; IF SC="0" THEN MANT:=TALLY;
  TALLY:=MANT; SI:=LOC FRAC; SI:=SI+7; IF SC GTR "0"
  THEN TALLY:=TALLY+1; TALLY:=TALLY+FRAC; MANT:=TALLY;
  SI:=T1; IF SC="1" THEN BEGIN
    DI:=A; DI:=DI+MANT; DI:=DI+2;
    SI:=TSI; DS:=4CHR;
    TALLY:=TALLY+4; MANT:=TALLY; END;
  SI:=LOC MANT; SI:=SI+6; DI:=A; DS:=2CHR;
  END;
  IF S:=ABS(R) GEQ MAXIM OR S LEQ 10*(-DIGITS) AND S NEQ 0 THEN
    WRITE(SCR[*],G,DIGITS,R) ELSE
    WRITE(SCR[*],F,DIGITS,R);
  ADJUST(A,SCR)
  END;
PROCEDURE STOREPSR;
BEGIN INTEGER I;
DELETE(WORKSPACE,0);
I:=STORESEQ(WORKSPACE,PSR,PSR SIZE*8);
COMMENT USED TO CALL WRAPUP;
END;
PROCEDURE RESCANLINE;
BEGIN ADDRESS:=ABSOLUTEADDRESS; EOB:=0; END;
PROCEDURE PROCESS(MODE); VALUE MODE; INTEGER MODE; FORWARD;
PROCEDURE MESSAGEHANDLER; FORWARD;
PROCEDURE FUNCTIONHANDLER; FORWARD;
PROCEDURE ERRORMESS(N,ADDR,R); VALUE N,ADDR,R; REAL R;
INTEGER N; REAL ADDR; FORWARD; COMMENT LINE 500000;
STREAM PROCEDURE SETFIELD(A,S,L,R); VALUE S,L,R;
BEGIN DI:=A; DI:=DI+S; SI:=LOC R; SI:=SI+8; L(SI:=SI-1);
DS:=L CHR;
END;
COMMENT: VALUE OF GETFIELD IS L CHARACTERS, STARTING AT J-TH
CHARACTER OF A, RIGHT-ADJUSTED. L MUST BE LEQ 8 AND
J MUST BE LESS THAN 64;
REAL STREAM PROCEDURE GETFIELD(A,S,L); VALUE S,L;
BEGIN SI:=A; SI:=SI+S; DI:=LOC GETFIELD; DI:=DI+8; L(DI:=DI-1);
DS:=L CHR;
END;
REAL PROCEDURE TOPLINE(ORD); VALUE ORD; INTEGER ORD;
BEGIN
  INTEGER STREAM PROCEDURE CON(A); VALUE A;
  BEGIN SI:=LOC A; DI:=LOC CON; DS:=8OCT END;
  ARRAY A(0:1); INTEGER I;
  I:=CONTENTS(ORD,SIZE(ORD)-1,A);
  TOPLINE:=CON(A(0))/10000
END;
BOOLEAN PROCEDURE FUNCTIONHEADER(SPECS,HADDR);
ARRAY SPECS(0); REAL HADDR;
BEGIN
  LABEL A,B,C;
  INTEGER P;
  DEFINE NOTE=HADDR.[24:24]:=ADDRESS#,P8=8*P+1#;
  FRR:=0;
  SPECS(0):=SPECS(1):=SPECS(2):=SPECS(3):=0;
  NOTE; HADDR.[1:23]:=GT1:=ADDRESS;
  IF SCAN AND IDENT THEN
    BEGIN
      TRANSFER(ACCUM,2,SPECS,1,7);
      NOTE;
      IF SCAN THEN
        IF LFTARROW THEN
          BEGIN
            SPECS(1):=1;
            SPECS(3):=1;
            TRANSFER(SPECS,1,SPECS,33,7);
            GT2:=ADDRESS;
            IF SCAN AND IDENT THEN
              BEGIN
                TRANSFER(ACCUM,2,SPECS,1,7);
                NOTE;

```

```

00342000
00343000
00343010
00344000
00345000
00346000
00347000
00348000
00349000
00350000
00351000
00352000
00353000
00354000
00355000
00356000
00356010
00356020
00356030
00356040
00357000
00358000
00358010
00358020
00359000
00360000
00361000
00361010
00361020
00361030
00361040
00361050
00361060
00361070
00361072
00361100
00362000
00362100
00362105
00362107
00362110
00362120
00362130
00362140
00362145
00362146
00362147
00362150
00362160
00362170
00362180
00362200
00362210
00362220
00362230
00362240
00362250
00362260
00362270
00500000
00500100
00500150
00500200
00500300
00500325
00500350
00500400
00500450
00500500
00500600
00500700
00500750
00500800
00500900
00501000
00501100
00501150
00501200
00501250
00501300
00501400
00501500
00501550

```

```

IF SCAN THEN
C: IF IDENT THEN
BEGIN
P:=(SPECS[3]:=SPECS[3]+1)+3;
TRANSFER(ACCUM,2,SPECS,P8,7);
SPECS[2]:=1;
NOTE;
IF SCAN THEN IF IDENT THEN
BEGIN SPECS[2]:=2;
P:=(SPECS[3]:=SPECS[3]+1)+2;
TRANSFER(SPECS,1,SPECS,P8+8,7);
TRANSFER(SPECS,P8,SPECS,1,7);
TRANSFER(ACCUM,2,SPECS,P8,7);

B: NOTE; IF SCAN THEN
A: IF SEMICOLON THEN IF SCAN THEN
IF IDENT THEN
BEGIN
P:=(SPECS[3]:=SPECS[3]+1)+3;
TRANSFER(ACCUM,2,SPECS,P8,7);
GO TO B;
END ELSE GO TO A
ELSE ELSE ELSE
END ELSE GO TO A
ELSE END
ELSE GO TO A ELSE
END ELSE ERRORMESS(ERR:=1,GT2,0)
END ELSE GO TO C

ELSE
END ELSE ERRORMESS(ERR:=SYNTAXERROR,GT1,0);
FUNCTIONHEADER:=ERR=0;
ADDRESS:=HADDR.[24:24];
END FUNCTIONHEADER;

INTEGER PROCEDURE DAYTIME(B); ARRAY B[0]; FORWARD;
COMMENT ON LINE 8014000, ARRAY B MUST HAVE LENGTH
AT LEAST 3 WDS;
PROCEDURE EDITLINE; FORWARD;
INTEGER PROCEDURE LENGTH(A,M); VALUE M; BOOLEAN M; ARRAY A[0];
FORWARD; COMMENT LINE 8007900;
BOOLEAN PROCEDURE LABELSCAN(L,K); VALUE K; INTEGER K;
ARRAY L[0]; FORWARD; COMMENT LINE 8013910;

PROCEDURE CHECKSEQ(SEQ,L,INC); REAL SEQ,L,INC; FORWARD;
COMMENT ON LINE 8040000;
PROCEDURE RELEASEARRAY(D); VALUE D; REAL D;
BEGIN COMMENT RELEASE PERMANENT STORAGE FOR THE ARRAY DESC D;
INTEGER K,J,PT;
ARRAY BLOCK[0:32]; %SEE MAXWORDSTORE, LINE 17260
ARRAY TEMP[0:1];
IF D.REF NEQ 0 THEN
BEGIN DELETE1(WS,D.DIMPTR);
K:=CONTENTS(WS,D.INPTR,BLOCK)-1;
DELETE1(WS,D.INPTR);
FOR J:=0 STEP 2 UNTIL K DO
BEGIN TRANSFER(BLOCK,J,TEMP,6,2);
PT:=TEMP[0]; DELETE1(WS,PT); END;
END;

END;
PROCEDURE TRANSFERSP(DIR,SP,L,B,M,N); VALUE DIR,N,M,L;
INTEGER DIR,N,M,L;
ARRAY SP[0:0],B[0];
BEGIN COMMENT
DIR= INTO; TRANSFER N WORDS FROM B[1] INTO SP[M]
(ACTUALLY SP[*],M) SINCE ARRAY ROW IS USUALLY THE ARG)
DIR= OUTOF (OPPOSITE);
STREAM PROCEDURE MOVER(DIR,SP,M,B,L,N); VALUE DIR,
L,M,N;
BEGIN LOCAL T;
SI:=LOC L; DI:=LOC T; DI:=DI+1; DS:=7CHR;
SI:=SP; T(16(SI:=SI+32)); L(SI:=SI+8); L:=SI;
SI:=LOC M; DI:=LOC T; DI:=DI+1; DS:=7CHR;
SI:=B; T(16(SI:=SI+32)); M(SI:=SI+8); M:=SI;
SI:=LOC N; DI:=LOC T; DI:=DI+1; DS:=7CHR;
SI:=LOC DIR; SI:=SI+7;
IF SC="0" THEN
BEGIN SI:=M; DI:=L
END ELSE
BEGIN SI:=L; DI:=M
END;
T(2(DS:=32WDS)); DS:=N WDS;
END;

```

0050160
0050170
0050180
0050185
0050190
0050200
0050205
0050210
0050220
0050225
0050230
0050240
0050250
0050255
0050260
0050261
0050262
0050263
0050264
0050265
0050266
0050267
0050268
0050269
0050270
0050280
0050290
0050300
0050310
0050320
0050450
0050455
0050460
0080181
0208000
0208001
0208002
0208003
0208004
0208005
0208006
0208007
0208008
0208009
0208010
0208020
0300050
0300051
0300052
0300053
0300053
0300054
0300055
0300056
0300057
0300058
0300058
0300059
0300060
0300061
0300100
0300110
0300120
0300130
0300140
0300145
0300150
0300160
0300170
0300180
0300190
0300200
0300210
0300211
0300212
0300213
0300214
0300215
0300216
0300217
0300218
0300219
0300220

```

INTEGER K;
WHILE N:=N-K GTR 0 DO
  MOVER(DIR,SP(L:=L+K)DIV SPRSIZE,*J,
  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,OP,TYPE,ACCUM);
    FORMROW(0,1,ACCUM,2,ACOUNT);
    NUMBERCON(T,LOC,FIELD,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,I);
  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;

```

```

03002210
030023
030024
030025
03002600
03002700
030028
030080
03008100
03008150
030082
030083
030084
03008500
03008510
030085
03008540
03008550
030085
030085
03008590
030086
030086
03008623
03008626
030086
030086
03008660
03008670
030086
03008700
03008702
030087
030087
03008720
030087
0300874
03008742
03008743
0300874
0300874
03008750
0300876
0300877
0300878
03008790
03008795
0300880
03008820
03008830
0300884
0300885
0300886
03008870
0300888
0300889
0300890
03008910
03008920
0300893
0300900
0302340
03100000
0310010
0310010
0310011
03100120
03100130
0310014
0310041
0310041
03100420
0310043
0310043

```

```

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

```

```

03100440
03100445
03100450
03100452
03100460
03100462
03100470
03100805
03100807
03100810
03100840
03100850
03100860
03100870
03100880
03100890
03100900
03100910
03100920
03100930
03100940
03100950
03100960
03100962
03100970
03100980
03100990
03100992
03101000
03101010
03101012
03101014
03101020
03101030
03101032
03101040
03101050
03101060
03101070
03101080
03101090
03101100
03101110
03101120
03101130
03101140
03101150
03101160
03101170
03101180
03101182
03101184
03101190
03101200
03101202
03101210
03101220
03101222
03101230
03101240
03101242
03101250
03101251
03101252
03101253
03101254
03101255
03101256
03101257
03101258
03101260
03101270
03101280
03101290
03101300
03101350
03101400
03101450
03101500
03101510
03101520
03101521
03101530

```

```

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(LASICONSTANT);
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[L],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[L],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.
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

```

```

03101550
03101600
03101650
03101700
03101710
03101750
03101760
03101765
03101770
03101780
03101790
03101800
03101850
03101860
03101870
03101900
03101950
03102000
03102050
03102100
03102150
03102200
03102250
03102350
03102390
03102392
03102400
03102420
03102430
03102440
03102442
03102450
03102460
03102500
03102510
03102550
03102600
03102650
03102700
03102750
03102800
03102850
03102900
03102925
03102950
03102960
03102961
03102962
03102965
03102970
03102980
03103000
03103050
03103100
03103180
03103200
03103225
03103250
03103300
03103325
03103326
03103330
03103350
03103400
03103425
03103430
03103450
03103500
03103550
03103600
03103610
03103620
03103630
03103640
03103650
03103700
03103750
03103800
03103850
03103900
03103950
03104000
03104050

```

FILLER:


```

UNTIL NOT(SCAN AND ERR=0); %DROP THRU WHEN INPUT FIN DR 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 ANDRATOR (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 RGTARENV AND T NEQ
        QUAD AND T NEQ QUAD AND T NEQ
        RGTBRACKETV THEN GO ATO
      ELSE GO OPERAN
    ELSE
      IF T=FUNCTION THEN
        IF VAR.OPTYPE GTR NILADIC THEN
          ATO:=ANDRATOR:=TYPE=OPERATOR
        ELSE GO OPERAN
      ELSE
        OPERAN:=ANDRATOR:=TYPE=OPERAND;
      END OF ANDRATOR;
  BEGIN 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=LFTARENV THEN RT
  ELSE IF T=FUNCTION AND VAR.OPTYPE LEQ MONADIC THEN RT;
  END OF RGTOPERAND;
  BOOLEAN VALID;
  INTEGER OTO;
  INTEGER RCT,N; REAL COLONCTR;
  LABEL STACKOPERAND,STACKFUNCTION;
  DEFINE PTO=L#;
  LABEL AROUND, NOK,OK, LFTARROWL, LFTARENL, RGTARENL,
  SLASH,EXPL,ROTL,MONADICL,DYADICL,ERRL,SortL,
  SEMICOLONL,QUADL,DTL,RELATIONL,
  LFTBRACKETL,RGTBRACKETL,QUOTEQUADL;
  SWITCH OPERATORSWITCH:= % IN GROUPS OF 5, STARTING AT 1
    NOK,NOK,NOK,LFTARROWL,%1-4
    MONADICL,SLASH,OK,LFTARENL,RGTARENL,%5-9
    QUADL,LFTBRACKETL,RGTBRACKETL,ERRL,QUOTEQUADL,%10-14
    SEMICOLONL,OK,DTL,OK,OK,%15-19
    OK,DYADICL,DYADICL,MONADICL,RELATIONL,%20-24
    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 ANDRATOR(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 ANDRATOR(INFIX[I-1],OPERAND) THEN GO OK;
        T:=INFIX[I];
        T.LOCFIELD:=RDTATE;
        T.OPTYPE:=IF INFIX[I].OPTYPE NEQ DYADIC THEN MONADIC ELSE DYADIC;
        INFIX[I]:=T; GO TO STACKFUNCTION;
      EXPL:
      SLASH:
        BEGIN DEFINE STARTSEGMENT=#; %////////////////////
        IF INFIX[I-1].TYPEFIELD=FUNCTION THEN GO ERRL ELSE
        IF ANDRATOR(INFIX[I-1],OPERATOR) THEN
          BEGIN
            INFIX[I].LOCFIELD:=IF INFIX[I].LOCFIELD=SLASHV THEN
            REDUCT ELSE SCANV;

```

031040
031041
031042
031043
031044
031045
031046
031047
031048
031049
031050
031051
031052
031053
031054
031055
031056
031057
031058
031059
031060
031061
031062
031063
031064
031065
031066
031067
031068
031069
031070
031071
031072
031073
031074
031075
031076
031077
031078
031079
031080
031081
031082
031083
031084
031085
031086
031087
031088
031089
031090
031091
031092
031093
031094
031095
031096
031097
031098
031099

```

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,RGTPAREN,RTBRACKETV);
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,RTBRACKETV,RGTPAREN);
IF DTOP=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 SUBSCRIBED 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[NDCT]:=COLONCTR+1; % STORE NUMBER OF DIMENSIONS
N:=GETSPACE(1); % BUILD A DESCRIPTOR FOR # OF DIMENSIONS
T.SP:=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 REPLACEDP
INFIX[I-1].TYPEFIELD:=REPLACELUC;
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:

```

```

03104839
03104840
03104841
03104842
03104843
03104844
03104845
03104846
03104847
03104848
03104849
03104850
03104851
03104852
03104853
03104854
03104855
03104856
03104857
03104858
03104859
03104860
03104861
03104862
03104863
03104864
03104865
03104866
03104867
03104868
03104869
03104870
03104871
03104872
03104873
03104874
03104875
03104876
03104877
03104878
03104879
03104880
03104881
03104882
03104883
03104884
03104885
03104886
03104887
03104888
03104889
03104890
03104891
03104892
03104893
03104894
03104895
03104896
03104897
03104898
03104899
03104900
03104901
03104902
03104903
03104904
03104905
03104906
03104907
03104908
03104909
03104910
03104911
03104912
03104913
03104914
03104915
03104916
03104917
03104918
03104919
03104920
03104921
03104922
03104923
03104924
03104925
03104926
03104927
03104928
03104929
03104930
03104931
03104932
03104933
03104934
03104935
03104936
03104937
03104938
03104939
03104940
03104941
03104942
03104943
03104944
03104945
03104946
03104947
03104948
03104949
03104950
03104951
03104952
03104953
03104954
03104955
03104956
03104957
03104958
03104959
03104960
03104961
03104962
03104963
03104964
03104965
03104966
03104967
03104968
03104969
03104970
03104971
03104972
03104973
03104974
03104975
03104976
03104977
03104978
03104979
03104980
03104981
03104982
03104983
03104984
03104985
03104986
03104987
03104988
03104989
03104990
03104991
03104992
03104993
03104994
03104995
03104996
03104997
03104998
03104999
03105000
03105001
03105002
03105003
03105004
03105005
03105006
03105007
03105008
03105009
03105010
03105011
03105012
03105013
03105014
03105015
03105016
03105017
03105018
03105019
03105020
03105021
03105022
03105023
03105024
03105025
03105026
03105027
03105028
03105029
03105030
03105031
03105032
03105033
03105034
03105035
03105036
03105037
03105038
03105039
03105040
03105041
03105042
03105043
03105044
03105045
03105046
03105047
03105048
03105049
03105050
03105051
03105052
03105053
03105054
03105055
03105056
03105057
03105058
03105059
03105060
03105061
03105062
03105063
03105064
03105065
03105066
03105067
03105068
03105069
03105070
03105071
03105072
03105073
03105074
03105075
03105076
03105077
03105078
03105079
03105080
03105081
03105082
03105083
03105084
03105085

```

```

IF OTOP LSS ITOP DIV 2 THEN ELSE ERR:=SYNTAXERROR;
OPERATORS[OTOP:=OTOP+1]:=INFIX[I];
GO AROUND;
RGTBRACKETL: BEGIN DEFINE STARTSEGMENT=#; %////////////////////
BCT:=BCT+1;
IF OTOP+2 GEQ ITOP THEN
BEGIN
ERR:=SYNTAXERROR;
GO AROUND;
END;
OPERATORS[OTOP:=OTOP+1]:=COLONCTR;
GT1:=OPERATORS[OTOP:=OTOP+1]:=INFIX[I]; COLONCTR:=0;
IF I NEQ ITOP THEN
IF GT1.OPTYPE NEQ 3 THEN
OPERATORS[OTOP].OPTYPE:=IF RGTOPERAND(INFIX[I+1]) THEN
0 ELSE 2
ELSE
ELSE OPERATORS[OTOP].OPTYPE:=2;
IF J NEQ 0 AND INFIX[I-1].LOCFIELD=SEMICOLONV THEN
BEGIN
T.LOCFIELD:=BUILDNULL(LASTCONSTANT);
T.TYPEFIELD:=CONSTANT;
L:=L+1; K:=I;
SP[LOC]:=T;
END;
GO AROUND; END;
LFTARROWL:
IF I=1 THEN ERR:=SYNTAXERROR
ELSE
IF T:=INFIX[I-1].TYPEFIELD=OPERAND OR T=LOCALVAR THEN
INFIX[I-1].TYPEFIELD:=REPLACELOC
ELSE
IF T=OPERATOR THEN
IF T:=INFIX[I-1].LOCFIELD=QUAD OR T=QUADLFTARROW THEN
INFIX[I:=I-1].LOCFIELD:=QUADLFTARROW
ELSE IF T=RGTBRACKETV THEN INFIX[I-1].OPTYPE:=3
%WILL TEST LATER TO INDICATE REPLACEMENT IN MATRIX 3105154
ELSE ERR:=SYNTAXERROR
ELSE ERR:=SYNTAXERROR;
IF ERR=0 THEN GO OK ELSE GO AROUND;
QUOTEQUADL:
QUADL:
COMMENT INPUT IS BEING REQUESTED;
GO TO STACKOPERAND;
DOTL: BEGIN DEFINE STARTSEGMENT=#; %////////////////////
IF I GTR 2 THEN
IF (T:=INFIX[I-1]).TYPEFIELD=OPERATOR AND
ANDRATOR(T,OPERATOR) THEN
IF (T:=INFIX[I+1]).TYPEFIELD=OPERATOR AND
ANDRATOR(T,OPERATOR) THEN
IF ANDRATOR(INFIX[I-2],OPERAND) THEN
COMMENT THEN SYNTAX OK;
BEGIN
COMMENT STACK OPERATORS SO THAT IF GIVEN A+.XB
POLISH IS BA.+X;
OPERATORS[OTOP].OPTYPE:=TRIADIC;
OPERATORS[OTOP:=OTOP+1]:=INFIX[I-1];
INFIX[I].OPTYPE:=TRIADIC;
OPERATORS[OTOP:=OTOP+1]:=INFIX[I];
I:=I-1;
VALID:=TRUE;
END;
IF NOT VALID THEN ERR:=SYNTAXERROR;
VALID:=FALSE;
GO AROUND; END;
SEMICOLONL: BEGIN DEFINE STARTSEGMENT=#; %////////////////////
IF BCT NEQ 0 THEN
BEGIN
COLONCTR:=COLONCTR+1;
IF I-1=0 THEN ERR:=SYNTAXERROR
ELSE
BEGIN
UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTBRACKETV,RGTPAREN);
IF J NEQ 0 AND (T:=INFIX[I-1].LOCFIELD=SEMICOLONV
OR T=LFTBRACKETV) THEN BEGIN
T.LOCFIELD:=BUILDNULL(LASTCONSTANT);
T.TYPEFIELD:=CONSTANT;
L:=L+1; K:=I;
SP[LOC]:=T;
END;
END
END
ELSE COMMENT MUST BE MIXED MODE EXPRESSION;

```

```

03105087
03105090
03105100
03105115
03105130
03105132
03105134
03105136
03105138
03105140
03105145
03105150
03105152
03105154
03105156
03105158
03105159
03105160
03105161
03105163
03105165
03105167
03105169
03105171
03105173
03105175
03105176
03105180
03105182
03105184
03105186
03105188
03105190
03105192
03105194
03105195
03105196
03105197
03105198
03105200
03105202
03105204
03105205
03105206
03105207
03105208
03105209
03105210
03105211
03105212
03105213
03105214
03105215
03105216
03105217
03105218
03105219
03105220
03105221
03105222
03105223
03105224
03105225
03105226
03105227
03105228
03105229
03105230
03105231
03105232
03105233
03105234
03105235
03105236
03105237
03105238
03105239
03105240
03105241
03105242
03105243
03105244
03105245
03105246
03105247
03105248
03105249
03105250
03105251
03105252
03105253
03105254
03105255
03105256
03105257
03105258
03105259
03105260
03105261
03105262
03105263
03105264
03105265
03105266
03105267
03105268
03105269
03105270
03105271
03105272
03105273
03105274
03105275
03105276
03105277
03105278
03105279
03105280
03105281
03105282
03105283
03105284
03105285
03105286
03105287
03105288
03105289
03105290
03105291
03105292
03105293
03105294
03105295
03105296
03105297
03105298
03105299
03105300
03105301
03105302
03105303
03105304
03105305
03105306
03105307
03105308
03105309
03105310
03105311
03105312
03105313
03105314
03105315
03105316
03105317
03105318
03105319
03105320
03105321
03105322
03105323
03105324
03105325
03105326
03105327
03105328
03105329
03105330
03105331
03105332
03105333
03105334
03105335
03105336
03105337
03105338
03105339
03105340
03105341
03105342
03105343
03105344
03105345
03105346
03105347
03105348
03105349
03105350
03105351
03105352
03105353
03105354
03105355
03105356
03105357
03105358
03105359
03105360
03105361
03105362
03105363
03105364
03105365
03105366
03105367
03105368
03105369
03105370
03105371
03105372
03105373
03105374
03105375
03105376
03105377
03105378
03105379
03105380
03105381
03105382
03105383
03105384
03105385
03105386
03105387
03105388
03105389
03105390
03105391
03105392
03105393
03105394
03105395
03105396
03105397
03105398
03105399
03105400

```

```

BEGIN
  IF ANDORATOR(T:=INFIX[I-1],OPERATOR) THEN
    IF T.LOCFIELD NEQ SEMICOLONV THEN GO ERR1;
  UNSTACK(SP,PTOP,OPERATORS,OTOP,1,RGTPARENV,RGTBRACKETV);
  OPERATORS[OTOP:=OTOP+1]:=INFIX[I];
  END;
  GO AROUND;
END;
NOK:
ERR:=SYSTEMERROR;
GO AROUND;
ERR1:
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 ERR1
ELSE
ELSE
IF I=1 AND I NEQ ITOP THEN
IF RGTOPERAND(INFIX[I+1]) THEN GO ERR1
ELSE
ELSE
IF ANDORATOR(INFIX[I-1],OPERAND) OR RGTOPERAND(INFIX[I+1])
THEN
IF INFIX[I-1].LOCFIELD=RGTBRACKETV THEN
ELSE GO ERR1;
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;

```

```

031053383
031053385
031053390
031053395
031053400
031053403
031053405
031053407
031053655
031053660
031053661
031053662
031053663
031053665
031053668
031053669
031053670
031053671
031053672
031053673
031053674
031053675
031053677
031053680
031053682
031053685
031053700
031053710
031053715
031053717
031053720
031053722
031053724
031053726
031053728
031053730
031053732
031053734
031053736
031053738
031053740
031053742
031053744
031053746
031053748
031053750
031053751
031053752
031053754
031053758
031053760
031053762
031053764
031053766
031053768
031053770
031053772
031053773
031053774
031053775
031053800
031053820
031053835
031053850
031053900
031053950
031053952
031053960
031060000
031060001
031060050
031061000
031061500
031062000
031062500
03106252
03106253
03106254
03106255
03106256
03106258
03106260
03106262

```

```

T.OPTYPE:=MONADIC;
T.LOCFIELD:=QUADLFTARROW;
SP[LOC]:=1;
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 YTOP+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 PRG 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(CACCUM,2,GTA,0,7);
IF(CIF 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[L];
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&DDPNV[CDID];
SP[INDC]:=GTA[L];
END ELSE %IT MUST BE A VECTOR
SP[LOC]:=GTA[L];
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 NRDWS:=(COLDROW:=NROWS)+K:=ENTIER(LENGTH/
SPRSIZE+1)

```

```

03106261
03106261
03106261
03106271
03106301
03106351
03106401
03106451
03106501
03106521
03106531
03106551
03106601
03106651
03106701
03106751
03106801
03106851
03106901
03106951
03107001
03107051
03107101
03107151
03107201
03107221
03107251
03107301
03107351
03107401
03107451
03107501
03107551
03108001
03108021
03108031
03108041
03108061
03108081
03108101
03108121
03108141
03108161
03108201
03108221
03108241
03108261
03108281
03108301
03108321
03108341
03108361
03108381
03108401
03108421
03108441
03108461
03108481
03108501
03108521
03108541
03108561
03108601
03110001
03110101
03110201
03110211
03110221
03110301
03110311
03110401
03110411
03110501
03110601
03110701
03110711
03110721
03110731
03110801
03110901
03110911
03110911

```

```

        BEGIN COMMENT TAKE EASY WAY OUT FOR NOW;
        GETSPACE:=-1@10; %CAUSES INVALID INDEX
        NROWS:=OLDRROW; ERR:=SPERROR;
        GO TO ENDGETSPACE
        END;
        K:=K*SPRSIZE;

        L:=LASTAREA;
        IF OLDRROW = -1 THEN COMMENT FIRST ROW OF SP;
            BEGIN SP[0,0].NEXT:=L:=1; K:=K-1
            END ELSE
        BEGIN SP[LOC].NEXT:=(OLDRROW+1)*SPRSIZE;
            L:=(OLDRROW+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&DDPNVW[CDID]&LASTCONSTANT[CLOC];
                        LASTCONSTANT:=L;
                    END ELSE COMMENT VECTOR;
                    BEGIN L:=K:=GS(ATOP+1);
                        TRANSFERSP(INTO,SP,L+1,A,1,ATOP);
                        SP[LOC]:=ATOP;

```

```

031110930
031110930
031110930
031110930
031110960
031111030
031111130
031111230
031111300
031111400
031111500
031111600
031111700
031111800
031111900
031112000
031112100
031112200
031112300
031112400
031112500
031112600
031112700
031112800
031112900
031113000
031113010
031113100
031113200
031113300
031113400
031113500
031113600
031113610
031113700
031113800
031113900
031114000
031114010
031114100
031114200
031114300
031114310
031114400
031114500
031114520
031114530
031114540
031114550
031114560
031114570
031114580
031114590
031114600
031114610
031114620
031114630
031114640
031114650
031114660
031114670
031114680
031114690
031114700
031114705
031114710
031114720
031114730
031114740
031114750
031114760
031114770
031114780
031114790
031114800

```

```

BUILDCONSTANT:=L:=GS(1); %VECTOR DESCRIPTOR
SPLOC:=K&1[CRF]&DDPNVW[CDID]&LASTCONSTANT[CLOC];
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]#,
BRFG=SP[STMINUS]#,
BACKPT=6:36:12#,
CI=18:36:12#,
SPTSP=30:30:18#,
PRGMKS=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:=DEPTHERORR;
PROCEDURE POP;
BEGIN REAL U;
IF ST GIR 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

I:=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;

```

```

0311148
0311148
0311148
0311148
0311148
0311148
0311149
0311150
0311151
0311152
0311153
0311154
0311155
0311156
0311157
0311158
0311159
0311159
0311159
0311159
0311159
0311159
0311159
0311160
0311161
0311161
0311162
0311162
0311170
0311171
0311172
0311173
0311173
0311174
0311175
0311175
0311176
0311176
0311176
0311176
0311177
0311178
0311179
0311179
0311180
0311181
0311182
0311183
0311184
0311186
0311186
0311187
0311187
0311188
0311188
0311189
0311190
0311190
0311191
0311191
0311192
0311193
0311194
0311195
0311196
0311197
0311198
0311198
0311199
0311199
0312000
0312010
0312020
0312021
0312030
0312040
0312050
0312060
0312070
0312080
0312090
0312100
0312110
0312115
0312120
0312125

```



```

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*X8);
    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*X8);
      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(I, 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);

```

03121235
03121236
03121237
03121238
03121239
03121240
03121241
03121242
03121243
03121244
03121245
03121246
03121247
03121248
03121249
03121250
03121251
03121252
03121253
03121254
03121255
03121256
03121257
03121258
03121259
03121260
03121261
03121262
03121263
03121264
03121265
03121266
03121267
03121268
03121269
03121270
03121271
03121272
03121273
03121274
03121275
03121276
03121277
03121278
03121279
03121280
03121281
03121282
03121283
03121284
03121285
03121286
03121287
03121288
03121289
03121290
03121291
03121292
03121293
03121294
03121295
03121296
03121297
03121298
03121299
03121300
03121301
03121302
03121303
03121304
03121305
03121306
03121307
03121308
03121309
03121310
03121311
03121312
03121313
03121314
03121315
03121316
03121317
03121318
03121319
03121320
03121321
03121322
03121323
03121324
03121325
03121326
03121327
03121328
03121329
03121330
03121331
03121332
03121333
03121334
03121335
03121336
03121337
03121338
03121339
03121340
03121341
03121342
03121343
03121344
03121345
03121346
03121347
03121348
03121349
03121350
03121351
03121352
03121353
03121354
03121355
03121356
03121357
03121358
03121359
03121360
03121361
03121362
03121363
03121364
03121365
03121366
03121367
03121368
03121369
03121370
03121371
03121372
03121373
03121374
03121375
03121376
03121377
03121378
03121379
03121380
03121381
03121382
03121383
03121384
03121385
03121386
03121387
03121388
03121389
03121390
03121391
03121392
03121393
03121394
03121395
03121396
03121397
03121398
03121399
03121400
03121401
03121402
03121403
03121404
03121405
03121406
03121407
03121408
03121409
03121410
03121411
03121412
03121413
03121414
03121415
03121416
03121417
03121418
03121419
03121420
03121421
03121422
03121423
03121424
03121425
03121426
03121427
03121428
03121429
03121430
03121431
03121432
03121433
03121434
03121435
03121436
03121437
03121438
03121439
03121440
03121441
03121442
03121443
03121444
03121445
03121446
03121447
03121448
03121449
03121450
03121451
03121452
03121453
03121454
03121455
03121456
03121457
03121458
03121459
03121460
03121461
03121462
03121463
03121464
03121465
03121466
03121467
03121468
03121469
03121470
03121471
03121472
03121473
03121474
03121475
03121476
03121477
03121478
03121479
03121480
03121481
03121482
03121483
03121484
03121485
03121486
03121487
03121488
03121489
03121490
03121491
03121492
03121493
03121494
03121495
03121496
03121497
03121498
03121499
03121500

```

SETFIELD(NEWDESC,7,1,IF BOOLEAN(T.SCALAR)
THEN SCALARDATA ELSE ARRAYDATA);
MOVE(NEWDESC,1,OLDDDESC); K:=1;
IF (IF VARSIZE=0 THEN FALSE ELSE
K:=SEARCHORD(VARIABLES,NEWDESC,J,7)=0)
THEN BEGIN
K:=CONTENTS(VARIABLES,J,OLDDDESC);
DELETE1(VARIABLES,J);
IF GETFIELD(OLDDDESC,7,1)=ARRAYDATA THEN
RELEASEARRAY(OLDDDESC[1]);
END ELSE
BEGIN VARSIZE:=VARSIZE+1; J:=J+K-1;
MOVE(OLDDDESC,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;
```

```

PROCEDURE SPCOPY(S,D,N); VALUE S,D,N; INTEGER S,D,N;
BEGIN
INTEGER K;
WHILE (N:=N-K) GTR 0 DO
TRANSFERSPC(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[LDC]);
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[LDC]-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[MOC];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[MOC];
TRANSFER(GTA,1,ACCUM,2,7); OPERANDTOSYMTAB(N);
END
ELSE BEGIN SP[MOC]:=SP[LDC];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 FNDFUNC,TERMINATE,DONE;
LABEL BUMPLINE;
LABEL TRYNEXT;
REAL STREAM PROCEDURE CON(A); VALUE A;
BEGIN SI:=LOC A; DI:=LOC CON; DS:=8DEC;

```

0312465
0312470
0312471
0312480
0312485
0312490
0312495
0312500
0312505
0312510
0312515
0312517
0312518
0312520
0312521
0312525
0312530
0312535
0312536
0312537
0312537
0312537
0312540
0312540
0312545
0312550
0312555
0313000
0313010
0313020
0313030
0313040
0313050
0313060
0313100
0313110
0313120
0313130
0313140
0313150
0313160
0313170
0313200
0313210
0313220
0313230
0313240
0313250
0313259
0313260
0313270
0313280
0313300
0313305
0313310
0313315
0313320
0313330
0313335
0313340
0313345
0313350
0313355
0313360
0313365
0313366
0313367
0313368
0313370
0313375
0313380
0313500
0314000
0314002
0314005
0314005
0314005
0314005
0314006
0314007


```

T:=SP[NOC]; SP[NOC].NAMED:=1; N:=T;
END;
WHILE ST GEQ TLAST AND FRR=0 DO POP; %GET RID OF TEMPS
OLDDATA:=(T:=AREG).SPF; POP;% GET RID OF INTERRUPT MKS
IF FRR 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[L0C].SPF+1;M:=SP[L0C].SPF+L;

SP[M0C]:=0;SP[L0C].SPF:=(M:=M-1)-L;
COMMENT NOW INITIATE ANY OLD FUNCTIONS, AND GET POLISH
GOING;
LASTM:=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[M0C].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[M0C];
LNUM:=IF TAKE THEN SP[L0C] ELSE (PUT:=SP[L0C])-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 x 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;

```

```

0314001
0314002
0314003
0314004
0314005
0314006
0314007
0314008
0314009
0314010
0314011
0314012
0314013
0314014
0314015
0314016
0314017
0314018
0314019
0314020
0314021
0314022
0314023
0314024
0314025
0314026
0314027
0314028
0314029
0314030
0314031
0314032
0314033
0314034
0314035
0314036
0314037
0314038
0314039
0314040
0314041
0314042
0314043
0314044
0314045
0314046
0314047
0314048
0314049
0314050
0314051
0314052
0314053
0314054
0314055
0314056
0314057
0314058
0314059
0314060
0314061
0314062
0314063
0314064
0314065
0314066
0314067
0314068
0314069
0315000
0315001
0315002
0315003
0315004
0315005
0315006
0315007
0315008
0315009
0315000

```



```

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(CSP[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[CLOC];%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[I]:=SUBINDEX(MAP[I],SIZEMAP[I],PROGRESS[I])
xBLOCKSIZE[I];
FOR I:=2 STEP 1 UNTIL RANK DO
K:=POINTER[I]:=K+SUBINDEX(MAP[I],SIZEMAP[I],
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 % OUTF..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[CLOC];%STORE NEW DESC.
OLDDATA:=CHAIN(I,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;

```

03150650
03150660
03150662
03150663
03150664
03150665
03150666
03150667
03150669
03150670
03150680
03150690
03150700
03150710
03150712
03150714
03150720
03150730
03150740
03150750
03150760
03150770
03150780
03150800
03150805
03150810
03150815
03150820
03150825
03150830
03150835
03150840
03150845
03150850
03150855
03150860
03150865
03150870
03150875
03150876
03150878
03150880
03150882
03150886
03150887
03150888
03150890
03150895
03150900
03150901
03150902
03150904
03150905
03150906
03150908
03150909
03150910
03150912
03150915
03150920
03150925
03150930
03150932
03150940
03150942
03150944
03150946
03150950
03150960
03150962
03150970
03150971
03150972
03150974
03150980
03150990
03151000
03151010
03151020
03151030
03151040
03151300
03151305

```

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[NOI]; L:=L+1;
    END ELSE BEGIN % INTO
      SP[NOI]:=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])x
  BLOCKSIZE[LEVEL];%POINTER[ ] IS 0
  FOR I:=LEVEL+1 STEP 1 UNTIL RANK DO
    K:=POINTER[I]:=K+SUBINDEX(MAP[I],SIZEMAP[I],
    PROGRESS[I])xBLOCKSIZE[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 BOOLFAN(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 IMS(N); VALUE N; INTEGER N;
  BEGIN COMMENT N=0 FOR REGULAR INTERRUPT MKS
    N=1 FOR QUAD INTERRUPT MKS
    N=2 FOR QUAD INTERRUPT MKS
    N=3 FOR EXECUTION LINE FOLLOWING
    N=4 FOR SUSPENDED FUNCTION;
  INTEGER L,M;
  PUSH;AREG:=DLDDATA&(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,WDLIN,F,CC;
  COMMENT WDLIN=#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 (JxK);
  WDLIN := (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)xI)+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,WDLIN);
        FORMROW(3,CC,BUFFER,ENTIER(M MOD 8),NJ:=LINESIZE-CC);

```

03151310
03151320
03151322
03151330
03151340
03151350
03151360
03151420
03151430
03151440
03151450
03151450
03151460
03151470
03151480
03151482
03151484
03151490
03151500
03151510
03151520
03151550
03151552
03151554
03151556
03151557
03151558
03151560
03151562
03151564
03151566
03151568
03151570
03151580
03151800
03152000
03152003
03152006
03152010
03152100
03152110
03152120
03152122
03152124
03152150
03152155
03152160
03152170
03152180
03152190
03152195
03152200
03152210
03152220
03152225
03152230
031522500
031522510
031522512
031522514
031522515
031522516
031522517
031522518
031522520
031522530
031522540
031522550
031522560
031522570
031522580
031522590
031522595
031522600
031522610
031522615
031522625
031522630
031522635
031522640
031522644


```

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(SPILOC,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(O,L+LD.SPF,J:=FINDSIZE
(LD)) ELSE J:=MOVEN(O,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);
SPILOC:=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); SPI[MOC]:=T;
LASTMKS:=ST-STACKBASE;
PUSH; IF ERR=0 THEN
BEGIN AREG:=PROGMKS&LASTMKS[BACKPT]&1[CI]&M[SP TSP];
POLLOC:=M:=T.SPF; POLTOP:=SPI[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:=SPI[MOC]).DID=IMKS

```

```

03152646
03152648
03152650
03152655
03152660
03152670
03152680
03152690
03152700
03152710
03152720
03153000
03153010
03153020
03153025
03153030
03153032
03153034
03153036
03153038
03153040
03153042
03153050
03153060
03153070
03153080
03153082
03153090
03153100
03153110
03153150
03153160
03153161
03153162
03153170
03153180
03153190
03153200
03153210
03153220
03153225
03153300
03153310
03153320
03153330
03153340
03153350
03153360
03153370
03153380
03153381
03153382
03153390
03153400
03153410
03153420
03153430
03153500
03153510
03153520
03153530
03153540
03153550
03153560
03153570
03153580
03153590
03153600
03153610
03153620
03153630
03154000
03154100
03154200
03154300
03154400
03154500
03154600
03154700
03154710
03154800
03154850
03154900

```

```

AND N.QUADIN=4) THEN POPPROGRAM:=TRUE;
IF A.DID NEQ FMKS THEN
FORGET PROGRAM(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[MOC].SUSPENDED) THEN BEGIN SP[MOC].RF:=L:=
SP[MOC].RF-1; IF L=0 THEN SUSPENSION:=0; END;
SP[MOC]:=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:BUFSIZE);
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:=R(0) GEQ 1 THEN COMMENT A VECTOR;
BEGIN
L:=GETSPACE(N:=(R-1)DIV 8+2);
TRANSFER SP(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(MOC):=R;
COMMENT WE HAVE BUILT THE VECTOR AND DESCRIPTOR;
BUILDALPHA:=N
END
ELSE BEGIN BUILDALPHA:=1; ADDRESS:=R END;

```

```

03155000
03155090
03155100
03155200
03155300
03155400
03155500
03155600
03155700
03155800
03155900
03156000
03156100
03156200
03156300
03156400
03210000
03210005
03210010
03210020
03210030
03210040
03210050
03210060
03210065
03210070
03210080
03210085
03210090
03210100
03210110
03210120
03210130
03210140
03210150
03210160
03210170
03210180
03210190
03210200
03210210
03210220
03210224
03210226
03210228
03210230
03210240
03210250
03210260
03210270
03210280
03210290
03210300
03210310
03210320
03210330
03210340
03210350
03210360
03210370
03210380
03210385
03210390
03210400
03210410
03210420
03210430
03210432
03210440
03210450
03210460
03210470
03210480
03210482
03210484
03210490
03210492
03210495
03210497
03210500
03210510
03210520
03210521

```

```

      %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[LOC]; 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[NO];
  N:=N-1; ROW := SP[NO];
  TOP := SIZE DIV (MAT:=ROW*COL);
  NEWDESC := DESC;
  NEWDESC.SPF := M := GETSPACE(SIZE+RANK);
  SPCOPY(L,M,RANK-2);
  N:=M+RANK-1; SP[NO]:=ROW;
  N:=N-1; SP[NO]:=COL;
  J:=0; M:=M+RANK;
  WHILE J LSS TOP DO BEGIN

```

0321052
0321053
0321060
0321061
0321062
0321063
0321064
0321065
0321066
0321067
0321068
0321069
0321070
0321071
0321072
0321073
0321074
0321075
0321076
0321077
0321078
0321079
0321080
0321081
0321082
0321083
0321084
0321084
0321085
0321086
0321087
0321088
0321089
0321090
0321091
0321092
0321092
0321093
0321094
0321095
0321096
0321097
0321098
0321098
0321099
0321100
0321101
0321102
0321103
0321104
0321105
0321106
0322000
0322010
0322010
0322011
0322011
0322011
0322020
0322030
0322032
0322033
0322033
0322033
0322034
0322035
0322036
0322037
0322050
0322060
0322065
0322066
0322070
0322080
0322090
0322095
0322100
0322101

```

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[L0C] 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 := (((((((((0.0016063118 × 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;

```

```

0322102
0322110
0322120
03221300
03221350
0322140
0322140
03221410
03221420
03221500
0322400
0322500L
03225100
03225200
03225300
03225400
03225500
03225600
03225700
03225800
03225900
03226000
03226100
03226200
03226300
03226400
03226600
03226700
03226800
03226900
03227000
03227100
03227200
03227300
03227305
03227310
03227315
03227320
03227321
03227325
03227330
03227335
03227340
03227345
03227350
03227355
03227360
03227365
03227367
03227370
03227375
03227380
03227800
03227810
03227900
03228550
03228600
03228605
03228607
03228610
03228615
03228620
03228625
03228630
03228635
03228640
03228645
03228650
03228655
03228700
03228800
03228900
03229000
03229100
03229200
03229400
03229500
03229600
03229700
03229800
03229900
03230000
03230010

```

OP	APL OPERATOR	OP	APL OPERATOR
0		0	FACT-COMB
1	+	1	LSS
2	TIMES	2	=
3	-	3	GEQ
4	DIV	4	GTR
5	*	5	NEQ
6	RNDM	6	LEQ
7	RESD=ABS	7	AND
8	MIN=FLR	8	OR
9	MAX=CEIL	9	NAND
	NOT	10	NDR
		11	LN=LOG
22	THE "CIRCLE" OPERATORS FOLLOW	22	SQRT(1-B*2)
23	PI * X	23	SIN
24	ARCTANH	24	COS
25	ARCCOSH	25	TAN
26	ARCSINH	26	SQRT(1+B*2)
27	SQRT(B+2-1)	27	SINH
28	ARCTAN	28	COSH
29	ARCCOS	29	TANH
	ARCSTN		

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 OF 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 FR:=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

```

03230015
03230020
03230025
03230030
03230035
03230040
03230045
03230050
03230055
03230060
03230061
03230062
03230063
03230064
03230065
03230066
03230067
03230068
03230069
03230070
03230071
03230072
03230073
03230074
03230075
03230080
03230085
03230090
03230100
03230200
03230210
03230300
03230302
03230303
03230305
03230310
03230320
03230330
03230340
03230345
03230350
03230355
03230357
03230360
03230400
03230500
03230600
03230700
03230800
03230900
03230905
03230910
03230920
03230923
03230925
03230930
03230935
03231000
03231010
03231100
03231200
03231300
03231400
03231500
03231600
03231700
03231800
03231900
03232000
03232100
03232200
03232300
03232400
03232500
03232510
03232600
03232700
03232800
03232900
03233000
03233100
03233200
03233300

```

ANS := IF RIGHT * LEFT = 1 THEN 0 ELSE 1; %XNAND
ANS := IF RIGHT + LEFT = 0 THEN 1 ELSE 0; %XNOR
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
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);

```

032233400
032233500
032233550
032233560
032233570
032233600
032233603
032233606
032233609
032233610
032233612
032233615
032233618
032233620
032233621
032233624
032233627
032233630
032233631
032233633
032233636
032233639
032233642
032233645
032233648
032233651
032233654
032233657
032233660
032233663
032233666
032233669
032233675
032233678
032233680
032233700
032233705
032233710
032233715
032233720
032233725
032233730
032233735
032233740
032233745
032233750
032233755
032233760
032233765
032233770
032233775
032233780
032233785
032233790
032233800
032233850
032233860
032233900
032233902
032233903
032233904
032233906
032233908
032233910
032234000
032234100
032234110
032234200
032234230
032234240
032234400
032234410
032234420
032234425
032234430
032234440
032234500
032234510
032234600
032234700
032234800
032234900
032235000


```

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[LDC],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[LDC],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 := DDPUVW;
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,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[LDC]; 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[LDC] := 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;

```

```

03235100
03235200
03235300
03235400
03235500
03235510
03235600
03235700
03235800
03235900
03236000
03236100
03236200
03236300
03236400
03236500
03236600
03236700
03236710
03236800
03236900
03237000
03237050
03237060
03237065
03237068
03237070
03237075
03237080
03237090
03237100
03237150
03237200
03237300
03237400
03237410
03237500
03237600
03237700
03237800
03237900
03238000
03238100
03238200
03238290
03238300
03238400
03238450
03238500
03238550
03238560
03238570
03238580
03238590
03238600
03238700
03238800
03238805
03238810
03238815
03238820
03238850
03238900
03238910
03238915
03238950
03239000
03239050
03239055
03239060
03239070
03239100
03239150
03239200
03239210
03239250
03239300
03239350
03239355
03239360
03239365
03239370
03239375

```



```

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 ERR:="--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 := DDPVW; 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;

```

03239380
03239385
03239390
03239395
03239400
03239405
03239410
03239415
03239420
03239425
03239430
03239435
03239440
03239445
03239450
03239455
03239460
03239465
03239470
03239475
03239500
03239550
03239600
03239605
03239610
03239615
03239620
03239625
03239630
03239632
03239635
03239638
03239641
03239644
03239647
03239650
03239653
03239656
03239659
03239662
03239665
03239668
03239671
03239674
03239677
03239725
03239726
03239727
03239728
03239730
03239732
03239734
03239736
03239737
03239738
03239739
03239740
03239742
03239743
03239744
03239745
03239746
03239748
03239750
03239752
03239760
03239800
03239850
03239900
03239950
03240000
03240050
03240100
03240150
03240152
03240155
03240160
03240200
03240750
03240800
03240802
03240805
03240807

```

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 LRRANK := 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+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 COMMAPP;
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;

```

```

0324081
0324082
0324083
0324084
0324085
0324086
0324087
0324088
0324089
0324090
0324091
0324092
0324093
0324094
0324095
0324096
0324100
0324100
0324102
0324103
0324104
0324105
0324105
0324105
0324107
0324108
0324110
0324112
0324113
0324115
0324117
0324118
0324120
0324122
0324130
0324140
0324150
0324160
0324165
0324170
0324180
0324190
0324200
0324210
0324220
0324225
0324230
0324240
0324250
0324270
0324285
0324290
0324300
0324310
0324320
0324325
0324330
0324340
0324350
0324354
0324354
0324354
0324354
0324355
0324360
0324370

```

```

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[NO] := 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 x
  IDENTITY := 0; %FOR -
  IDENTITY := 1; %FOR DIV
  IDENTITY := 1; %FOR *
  IDENTITY := 0; %NO REDUCTION ON RNDM
  IDENTITY := 0; %FOR RESD
  IDENTITY := BIGGEST; %FOR MIN
  IDENTITY := -BIGGEST; %FOR MAX
  IDENTITY := 1; %NOT ISNT DYADIC
  IDENTITY := 0; %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 x 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 := DPUVW;
  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

```

```

03243705
0324371
0324372
0324373
0324374
0324375
0324380
0324390
0324400
0324410
0324420
0324430
0324440
0324450
0324460
0324470
0324480
0324490
0324500
0324510
0324512
0324513
0324514
0324515
0324516
0324620
0324630
0324635
0324640
0324650
0324660
0324670
0324680
0324690
0324700
0324710
0324720
0324730
0324740
0324750
0324751
0324752
03247525
0324760
0324770
0324780
0324781
0324782
0324783
0325330
0325331
0325332
03253325
0325333
03253334
032533345
03253350
03253355
0325336
03253367
03253370
0325340
0325350
0325360
0325370
0325380
0325390
0325400
0325410
0325420
0325430
0325435
03254360
03254370
0325440
0325450

```

```

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 SPINOC)-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:=SPLOC=0 THEN BEGIN DESC:=NULLV; GO TO QUIT; END;
    IF TOP = 1 THEN BEGIN DESC.SPF := N := GETSPACE(2);
      DESC.RF := SPINOC := 1;
      N:=N+1; SPINOC:=SPIMOC; 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:=SPLOC;
  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:=SPINOC) 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 x 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 SPINOC:=SUM ELSE SPINOC:=SPIMOC;
  N:=N+1; M:=M+1; END;
T := GET(ALONG, RANK);
FACTOR := 1; TOP := RIGHT+ALONG;
FOR N:=RIGHT+RANK-1 STEP -1 UNTIL TOP DO FACTOR:=
  FACTOR x SPINOC;
N:=RIGHT + RANK - 1; DIM := SPINOC;
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 SPLOC=1 THEN FOR K:=1 STEP 1 UNTIL FACTOR DO BEGIN
    SPIMOC:=SPINOC; I:=I+1; M:=M+1; N:=N+1;
    END ELSE BEGIN I:=I+FACTOR; N:=N+FACTOR; END;
  GO TO QUIT;
RANK: 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;
PDP;
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;
BBOOLEAN 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

```

```

03254511
03254600
03254700
03254800
03254810
03254900
03255200
03255300
03255400
03255500
03255600
03255700
03255800
03255805
03255810
03255815
03255820
03255825
03255830
03255835
03255850
03255855
03255860
03255900
03255800
03255900
032556900
032556910
032556920
032556930
032557000
032557100
032557120
032557130
032557132
032557135
032557138
032557140
032557150
032557160
032557200
032557300
032557400
032557410
032557500
032557600
032557650
032557700
032557800
032557900
03258000
03258100
03258150
03258200
03258300
03258400
03258500
03259300
03259500
03259600
03259900
03260000
03260100
03260150
03260200
03268020
03268040
03268060
03268080
03268100
03268120
03268140
03268160
03268180
03268200
03268220
03268220
03268240
03268250
03268260
03268270

```

```

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+1, SUM) THEN GO TO DOMAIN;
IF SUM NEQ DIM THEN BEGIN ERR:=RANKERROR; GO TO QUIT; END;
IF RANK=0 THEN BEGIN
DIM:=SP[NOC]; DESC.SPF:=N:=GETSPACE(LSIZE+1);
DESC.RF:=I; DESC.DID:=(IF I=0 THEN DDPUSW ELSE DDPUVW);
SPCOPY(L,N,1); L:=L+1; N:=N+1; 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;

```

```

03268280
03268300
03268320
03268330
03268340
03268350
03268360
03268380
03268400
03268420
03268440
03268443
03268445
03268447
03268449
03268451
03268453
03268456
03268458
03268460
03268480
03268500
03268520
03268540
03268580
03268600
03268620
03268640
03268660
03268680
03268700
03268720
03268740
03268760
03268780
03268800
03268820
03268840
03268860
03268880
03268900
03268903
03268906
03268910
03268920
03268940
03268960
03268980
03269000
03269020
03269040
03269060
03269080
03269100
03269120
03269140
03269160
03269180
03269190
03269200
03269220
03269240
03269250
03269260
03269280
03269360
03269370
03269380
03269400
03269420
03269440
03269460
03269480
03269500
03269520
03269540
03269560
03269580
03269600
03269620
03269640
03269660
03269800

```

```

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[LDC];
IF FINDSIZE(LARG)=1 THEN % A 1 ELEMENT VECTOR
BEGIN
L:=L+LARG.RF;
LARG.SCALAR:=1;
LEFT:=SP[LDC];
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.RE=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[LDC]*(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[LDC];
T:=T+SP[MOC]*E;
END;
OUTE:
L:=GETSPACE(1);
SP[LDC]:=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;

```

```

032269860
032269870
032269880
032269900
032269910
032269920
032269930
032269940
032269960
032269980
032269982
032269984
032269986
032269987
032269988
032269990
032269992
032269994
032269996
032269998
032270000
032270002
032270004
032270006
032270008
032270010
032270018
032270020
032270025
032270030
032270035
032270037
032270040
032270060
032270100
032270120
032270140
032270160
032270180
032270200
032270300
032270320
032270340
032270380
032270400
032270420
032270440
032270460
032270480
032270500
032270520
032270540
032270560
032270580
032270600
032270620
032270640
032270660
032270680
032270700
032270702
032270704
032270706
032270710
032270712
032270716
032270720
032270740
032270760
032270800
032270820
032270880
032270900
032270920
032270925
032270930
032270940
032270950
032270960
032270980
032271000

```

```

RIGHT:=SP[MOC]; % VALUE OF SCALAR
LEFT:=SP[LDC]; % 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[LDC] 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,LSIZE,RSIZE,SIZE,LL,MM,I,
  RROW,RCOL,LROW,LCOL,LJUMP,RJUMP,MSAVE,LSAVE,RSTART;
  REAL DESC,TFMP;
  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 x 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[LDC];
    N:=N-1; LROW:=SP[LDC]; END;
  IF LRANK=1 THEN BEGIN LROW:=1; LCOL:=SP[LDC]; END;
  IF RRANK=2 THEN BEGIN
    N:=M+RRANK-1; RCOL:=SP[LDC];
    N:=N-1; RROW:=SP[LDC]; 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 x 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;

```

03271020
03271040
03271060
03271080
03271100
03271120
03271140
03271160
03271180
03271200
03271220
03271240
03271260
03271280
03271300
03271320
03271340
03271360
03271380
03271400
03271420
03271440
03271460
03271480
03271500
03271520
03271540
03271560
03271580
03271600
03271800
03271820
03271840
03271860
03271880
03271900
03271920
03271940
03271960
03271965
03271970
03271975
03271980
03271982
03271985
03271990
03271992
03272000
03272001
03272002
03272003
03272004
03272005
03272006
03272009
03272040
03272045
03272050
03272060
03272070
03272080
03272100
03272110
03272120
03272140
03272142
03272145
03272150
03272155
03272160
03272170
03272175
03272180
03272185
03272240
03272245
03272360
03272380
03272400
03272420
03272440
03272460
03272480


```

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[LDC],SP[MOC],1,ROP,SP[NDC])
          THEN GO TO FORGET ELSE FIRST := FALSE;
      END ELSE BEGIN
        IF NOT OPERATION(SP[LDC],SP[MOC],1,ROP,TEMP)
          THEN GO TO FORGET;
        IF NOT OPERATION(TEMP,SP[NDC],-1,LOP,SP[NDC])
          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[LDC],SP[MOC],1,ROP,SP[NDC]) THEN
      GO TO FORGET ELSE N:=N+1;
GO TO QUIT;
FORGET: FORGETSPACE(DESC.SPF,RANK+SIZE);
DOMAIN: FRR:=DOMAINERROR;
QUIT: IF CHARACTER THEN BEGIN
  FORGETSPACE(MSAVE , RRANK+RSIZE);
  FORGETSPACE(LSAVE , LRANK+LSIZE); END;
RESULT := 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[LDC] := 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[LDC];
  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[LDC]) * 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[LDC] 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;

```

```

03272500
03272520
03272540
03272560
03272580
03272600
03272620
03272630
03272640
03272660
03272680
03272700
03272720
03272740
03272760
03272780
03272800
03272820
03272840
03272860
03272880
03272900
03272920
03273060
03273080
03273100
03273120
03273140
03273160
03273180
03273200
03273220
03273240
03273260
03273280
03273285
03273300
03273320
03273340
03273380
03273400
03273420
03273440
03273442
03273444
03273446
03273448
03273450
03273452
03273454
03273456
03273458
03273460
03273462
03273464
03273466
03273468
03273470
03273472
03273474
03273476
03273478
03273482
03273484
03273486
03273488
03273490
03273492
03273494
03273496
03273498
03273500
03273520
03273540
03273560
03273568
03273580
03273600

```

```

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;
        REAL 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;

```

03273620
0327362
0327362
0327362
03273628
0327363
0327364
0327366
03273680
03273700
0327372
0327374
0327376U
03273800
0327382
0327383
0327384
03273860
03273880
0327390
0327392
0327394U
03273960
0327398
0327399
03274000
03274010
0327402
0327403
0327404U
03274060
0327406
0327407
0327408
03274100
03274102
0327410
03274101
03274107
03274108
0327411
0327411
03274115
03274117
0327412
0327414
03274160
03274165
03274180
03274200
03274220
03274240
03274260
0327428
03274301
03274320
03274360
03274381
03274401
03274420
03274440
03274450
03274460
03274461
03274464
03274466
03274466
03274461
03274470
03274472
03274474
03274477
03274480
03274500
03274520
03274540
03274560
03274580
03274600
03274620
0327462
0327462

```

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*SPLLOC; 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[LQC];
FOR L:=L+JUMP STEP JUMP UNTIL K DO
IF NOT OPERATION(SP[LQC],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[LQC];
FOR N:=N+JUMP STEP JUMP UNTIL K DO BEGIN
M:=N-JUMP; L:=L+JUMP;
IF NOT OPERATION(SP[MOC],SP[LQC],-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[LQC] 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;

```

0327461
0327462
0327463
0327464
0327465
0327466
0327467
0327468
0327469
0327470
0327471
0327472
0327473
0327474
0327475
0327476
0327477
0327478
0327479
0327480
0327481
0327482
0327483
0327484
0327485
0327486
0327487
0327488
0327489
0327490
0327491
0327492
0327493
0327494
0327495
0327496
0327497
0327498
0327499
0327500
0327501
0327502
0327503
0327504
0327505
0327506
0327507
0327508
0327509
0327510
0327511
0327512
0327513
0327514
0327515
0327516
0327517
0327518
0327519
0327520
0327521
0327522
0327523
0327524
0327525
0327526
0327527
0327528
0327529
0327530
0327531
0327532
0327533
0327534
0327535
0327536
0327537
0327538
0327539
0327540
0327541
0327542
0327543
0327544
0327545
0327546
0327547
0327548
0327549
0327550
0327551
0327552
0327553
0327554
0327555
0327556
0327557
0327558
0327559
0327560
0327561
0327562
0327563
0327564
0327565
0327566
0327567
0327568
0327569
0327570
0327571
0327572
0327573
0327574
0327575
0327576
0327577
0327578
0327579
0327580
0327581
0327582
0327583
0327584
0327585
0327586
0327587
0327588
0327589
0327590
0327600
0327601
0327610
0327620
0327630
0327631
0327640
0327641
0327642
0327643
0327644
0327650
0327660
0327670
0327680
0327690


```

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[LDC];
L:=L-1;
ROW:=(IF M GTR 1 THEN SP[LDC] 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[MDC]:=SP[MDC]+1 GTR SP[NDC] THEN
BEGIN SP[MDC]:=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; %LDC 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[LDC]; %DESCRIPTOR FOR FUNCTION IS IN D
SEQ:=GETFIELD(D,FSQE-8,FFL); ORD:=GETFIELD(D,EPT-8,EFL);
N:=GETSPACE((M:=SIZE(ORD))x2+6); %GET SPACE FOR TABLE
SP[NDC]:=Mx2+5; %SIZE OF THE VECTOR WHICH FOLLOWS
D:=D&N[CSPF]&1[CRF]&0[BACKPT]; D.PRESENCE:=1;
SP[LDC]:=D; %THIS SETS UP THE FUNCTION DESCRIPTOR.
N:=N+1; SP[NDC]:=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[LDC]:=N+1x2-1;
END;
SP[MDC]:=GTA[0]; M:=M+1;
IF NUMERIC THEN SP[MDC]:=GTA[1] ELSE
BEGIN
IF SEQ:=GTA[1] LSS 0 THEN %RESULT OR ARG
BEGIN L:=N+SEQ+1; SP[LDC]:=I;
SEQ:=0;
END ELSE SEQ:=CON(SEQ)/10000;
SP[MDC]:=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[NDC]-2; %FIND LOCAL NAMES

```

0350014
0350014
0350014
0350015
0350015
0350016
0350020
0350021
0350030
0350031
0350032
0350033
0350035
0350040
0350045
0350050
0350055
0350060
0350065
0350070
0350071
0350075
0350080
0350085
0350085
0350090
0350110
0350120
0350130
0350140
0350160
0350161
0350162
0350163
0350170
0350180
0350190
0350200
0350210
0350220
0350230
0350240
0350250
0350260
0350270
0350271
0350280
0350290
0350300
0350310
0350320
0350330
0350340
0350350
0350360
0350370
0350380
0350400
0350410
0350420
0350430
0350431
0350440
0350450
0350460
0350470
0350480
0350490
0350500
0350510
0350600
0350610
0350620
0350630
0350640
0350650
0350660
0350670
0350680
0350690
0350700

```

FOR N:=N+4 STEP 2 UNTIL T DO %ONCE FOR EACH LOCAL
  BEGIN GT1:=SP[NOCL].[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;
    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[LOC]&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.SP[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;
  EXPDVR:=EXPVRL;
  INTOVR:=INTOVR;
  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;

```

COPY:

03507100
03507200
03507300
03507400
03507500
03507600
03507700
03507800
03507900
03508000
03508100
03508200
03508300
03508400
03508500
03508600
03508700
03508800
03508900
03509000
03509100
03509200
03509300
03509400
03509500
03510000
03510100
03510150
03510150
03510200
03510300
03510400
03510500
03510600
03609000
03609100
03609200
03609300
03609400
03700000
03700100
03700200
03700300
03700400
03700500
03700600
03700700
03700710
03700800
03700810
03700820
03700900
03700910
03700912
03700920
03700930
03700940
03700950
03700960
03701000
03701010
03701012
03701020
03701020
03701020
03701020
03701028
03701030
03701040
03701050
03701060
03701100
03701200
03701300
03701400
03701500
03701600
03701700
03701800
03701900
03702000
03702100
03702200

```

*----- 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 N1=N+4 STEP 2 UNTIL T DO
BEGIN ACCUM[0]:=SP[NOC];
FORMROW(0,1,ACCUM,1,7);
END;
TERPRINT; M:=M-1;
END;
END;% OF THE CASE STATEMENT
*-----END OF CASES-----
IF FALSE THEN EXECUTION;
BEGIN COMMENT EXECUTION LOOP;
INTEGER LOOP;
INTEGER INPUTIMS;
LABEL BREAKKEY;
LABEL SKIPPDP,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 %PONEY 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];

```

```

0370230
0370240
0370250
0370260
0370270
0370280
0370290
0370900
0370910
0370920
0370930
0370940
0370950
0370960
0370970
0370971
0370972
0370973
0370974
0370975
0370976
0370977
0370978
0370979
0370980
0370981
0370982
0370983
0370984
0370985
0370986
0370987
0370988
0371100
0371110
0375000
0375010
0375020
0375020
0375020
0375021
0375030
0375040
0375041
0375050
0375100
0375110
0375120
0375130
0375131
0375135
0375140
0375150
0375160
0375161
0375162
0375163
0375164
0375165
0375166
0375167
0375168
0375170
0375175
0375180
0375190
0375200
0375201
0375202
0375203
0375204
0375205
0375210
0375211
0375212
0375220
0375230
0375240
0375250
0375260

```



```

DEFINE STARTSEGMENT=#; %////////////////////////////////////
T:=AREG; L:=BREG,SPF;
IF BOOLEAN(T,SCALAR) THEN BEGIN ERR:=DOMAINERROR;GO TO SKIPPPOP;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 SKIPPPOP;
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 SKIPPPOP;
END;
END;

```

```

-----QUAD INPUT-----
EVALQQ: BEGIN LABEL EVALQQQUAD;
IF JUMP THEN BEGIN JUMP:=FALSE; GO TO EVALQQQUAD END;
CURRENTMODE:=INPUTMODE;
IMS(1); % SETUP MARKSTACKS FOR QUAD EXIT
IF ERR NEQ 0 THEN GO TO SKIPPPOP;
GO TO EXECEXIT;
EVALQQQUAD: % 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 SKIPPPOP;
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); %RESID/ABS
IF T.OPTYPE=MONADIC THEN GO TO XEQEPS ELSE MEMBER; %MEMBERSHIP
RHOP; %RHO
IDTAP; %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

```

03811002
0381101
0381101
0381101
03811014
0381101
0381101
0381102
03811030
03811040
0381105
0381105
0381106U
03811070
0381108
0381109
0381110
03811101
03811102
0381110
03811106
03811108
0381111
0381111
0381111
03811120
0381113
0381114
0381120
03812000
03813000
0381400
0381401
0381401
03814020
0381403
0381404
0381408
03814100
03814110
0381412
0381413
0381414U
03814150
0381416
0381417
03814180
03814500
0381500
0381600
0381700U
03817100
0381720
0381730
0381800
03819000
03820000
0382100
0382200
0382300U
03824000
0382500
0382600
0382700
03828000
0382900n
0383000
0383100
0383200U
03833000
0383400
0383500
0383600
03837000
0383800n
0383800
0383801
0383802U
03838030
0383804
0383900

```

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 SKIPPPOP;
  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-----
XEQPS: 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
  INITRUFF(BUFFER,MAXBUFFSIZE);RFSCANLINE;
  TRANSFERSP(OUTOF,SP,T.SPF+1,BUFFER,0,U);
  IF(U:=Ux8-M).GTR 0 THEN SETFIELD(BUFFER,M,U," ");
  IF T.SPF=0 OR NOT SCAN THEN RESULT:=OR&1[CRE]&DDPUVW[CDID]& NULL
  ELSE BEGIN POP;IF SETUPLINE THEN; GO TO SKIPPPOP;END
  END; END;
END; %-----END OF OPERATION ON STACK-----

```

03840000
03841000
03842000
03842100
03843000
03844000
03845000
03846000
03847000
03848000
03849000
03850000
03851000
03851100
03851110
03851115
03851120
03851140
03851160
03851180
03851200
03851220
03851240
03851260
03851300
03851310
03851400
03851500
03851600
03851610
03851620
03851630
03851700
03851720
03851760
03851780
03851880
03851890
03851892
03851894
03851896
03852000
03852020
03852040
03852060
03852080
03853000
03853005
03853010
03853015
03853020
03853030
03853040
03854000
03855000
03856000
03857000
03857010
03857020
03857040
03857060
03857080
03858000
03859000
03860000
03860010
03861000
03861010
03862000
03862005
03862010
03862020
03862030
03862032
03862040
03862042
03862044
03862050
03862052
03862060
03862070
03862080
03869960

SKIPP0P;

```

POP;POP;PUSH;IF ERR=0 THEN AREG:=RESULT;
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(MAXBUFSIZE,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
PPROGRAM(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);

```

03869970
03869980
03870000
03870100
03870110
03870200
03870210
03870220
03870300
03870400
03870500
03872000
03872100
03872200
03872210
03872300
03872400
03872410
03872500
03872600
03900000
03901000
03902000
03903000
03904000
03905000
03905010
03905020
03905030
03905035
03905040
03905042
03905044
03905046
03905048
03905050
03905052
03905053
03905054
03905056
03905058
03905060
03905070
03905080
03905082
03905084
03905086
03905088
03905090
03905095
03905100
03905200
03905203
03905205
03905206
03905207
03905210
03905215
03905300
03905310
03905400
03905600
03910000
03918100
03918200
03918201
03918209
03918210
03918220
03918250
03918300
03918400
03918450
03918455
03918456
03918458
03918459
03918460
03918462
03918464
03918465
03918467
03918470

```

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:=FUNCLC;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;
EXEEXIT: 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((NRDWS+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 I; DS:=2RESET; DS:=2SET;
DI:=B; MESSIZE(U:=DI; DI:=LOC I; 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

```

```

03918475
03918480
03918485
03918488
03918488
03918488
03918488
03918490
03918500
03918600
03919000
03919800
03919810
03919820
03919830
03919840
03919850
03919860
03919870
03919880
03919890
03919899
03919900
03919990
03919992
03920000
03920100
03920200
03950000
03950090
03950100
03950200
03951000
03951100
03951200
03951300
03951400
03951500
03951600
03951700
03951710
03951720
03951800
03951900
03952000
03952100
03952200
03952300
03952400
03952500
03952650
03952700
03952800
03952900
03953000
0395310
0395311
0395312
0395320
0395321
0395330
0395330
0395330
0395340
0396000
0500000
0500010
0500020
0500030
0500040
0500050
0500060
0500070
0500080
0500090
0500100
0500110
0500120
0500130
0500140
0500141
0500150

```

```

"1",
"5DEPTH",
"6DOMAIN",
"7EDITING",
"5INDEX",
"5LABEL",
"6LENGTH",
"5NONCE",
"4RANK",
"6SYNTAX",
"6SYSTEM",
"5VALUE",
"7SP FULL",
"7FLYKITE";
IF R NEQ 0 THEN
  BEGIN INDCENT(R); CHRCOUNT:=CHRCOUNT-1
  END;
FORMROW((IF R=0 THEN 2 ELSE 0),0,ERMES,Nx8+1,
ERMES[N], [1:5]);
FORMWD(0,"8 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;
LAREL EXEC, GUESS;
DEFINE T=GT1#, J=GT2#, I=GT3#;
PROCEDURE INITIALIZEPSR;
BEGIN FOR I:=0 STEP 1 UNTIL PSRSIZE-1 DO
  PSRMI[I] := 0;
  SEED:=STREAMBASE; ORIGIN:=1;
  FUZZ:=1@-11;
  LINESIZE:=72; DIGITS:=9;
END;
LADDRESS := ADDRESS := ABSOLUTEADDRESS;
WORKSPACE:=WORKSPACEUNIT;
ITEMCOUNT := EDB := 0;
IF NEXTUNIT=WORKSPACEUNIT THEN % ESTABLISH A WORKSPACE
  BEGIN
  WORKSPACE:=NEXTUNIT;
  SEQUENTIAL(WORKSPACE);
  INITIALIZEPSR;
  I:=STORESEQ(WORKSPACE,PSR,PSRSIZEx8);
  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,0,1);
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,
  FSTARTx8,7);
  CURLINE:=GT3:=TOPLINE(GT1:=FUNCPONTER);
  CHECKSEQ(CURLINE,GT3,INCREMENT); %GET INCREMENT
  CURLINE:=CURLINE+INCREMENT; INDENT(-CURLINE);
  FUNCSIZE:=SIZE(GT1);
  END;
  %-----INPUTMODE-----ERRORMODE-----
  GO TO EXEC; GO TO EXEC;
END;

```

```

05001510
05001511
05001512
05001540
05001600
05001601
05001602
05001700
05001710
05001800
05001820
05001830
05001900
05002000
05002010
05002100
05002110
05002130
05002200
05002210
05002220
05002300
05002310
05002400
07001000
07002000
07003000
07004000
07004100
07005000
07005010
07005020
07005030
07005040
07005050
07006000
07007000
07008000
07019000
07020000
07021000
07022000
07023000
07025000
07028000
07029000
07030000
07031000
07032000
07033000
07034000
07035000
07035900
07036000
07036010
07036020
07036030
07036040
07036050
07036060
07036070
07036080
07036090
07036100
07036110
07036120
07036130
07036140
07036150
07036160
07036170
07036180
07036190
07036100

```



```

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;
PROCEDURE APLMONITOR;
BEGIN
  REAL T;
  INTEGER I;
  BOOLEAN WORK;
  LABEL AROUND, NEWUSER;
  LABEL CALCULATE, EXECUTEIT, FUNCTIONSTART, BACKAGAIN;
  LABEL CALCULATEDIT;
  T := CUSER := 1;
  T := STATION;
  BEGIN FILL ACCUM[+] WITH "APL/B550", "0 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 RGT PAREN 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);

```

BACKAGAIN:

```

IF CURRENTMODE=CALCMODE THEN
  BEGIN INDENT(0); TERPRINT;
  IF NOT BOOLEAN(SUSPENSION) THEN
    BEGIN IF CURRENTMODE NEQ ERRORMODE THEN
      PROCESS(WRITEBACK);
      SPIO,0:=0; NROWS:=-1;

```

```

END;
CURRENTMODE:=CALCMODE;
END;
END;
IF EDITDG=1 THEN
  BEGIN MOVE(OLDBUFFER, BUFFSIZE, BUFFER);
  RESCANLINE; EDITDG:=0; GO TO CALCULATEDIT;
END;
I:=0;

```

07044001
07044005
07044010
07044100
07044200
07045000
07100000
07101000
07102000
07103000
07104000
07105000
07106000
07107000
07107100
07115000
07115533
07115534
07115535
07115536
07115538
07115539
07115540
07115542
07115550
07115560
07115570
07115600
07115700
07115800
07115900
07116000
07116100
07117000
07117100
07117400
07117500
07117600
07117700
07117800
07117900
07118000
07118100
07118200
07118300
07118400
07118410
07118500
07118510
07118600
07118610
07119000
07125000
07126000
07126010
07126020
07126100
07126200
07126300
07126310
07126320
07126321
07126322
07126323
07126324
07126326
07126330
07126332
07126333
07126334
07126335
07126336
07126337
07126338
07126340
07126341
07126342
07126350
07126360
07126370
07126380
07126390
07126400

```

GO AROUND;
EXECUTE IT;
PROCESS(XEQUATE); %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 RGT PAREN 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 C; 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(TSI:=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;
INTEGER I;
I:=LENGT(A,M,BUFFSIZE*8);
LENGTH:=IF M THEN I ELSE BUFFSIZE*8-I
END;
BOOLEAN 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;
STREAM PROCEDURE MOVEWDS(A,N,B); VALUE N;
BEGIN SI:=A; DI:=B; DS:=N WDS END;
INTEGER 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;

```

```

07127000
07128000
07129000
07129010
07129100
07130000
07131000
07131010
07131020
07131024
07131030
07131040
07131050
07131100
07132000
07133000
07134000
08007900
08007910
08008000
08008010
08008020
08008022
08008030
08008040
08008050
08008060
08008070
08008080
08008090
08008100
08008110
08008120
08008130
08008140
08008150
08008160
08008170
08008180
08008190
08008200
08008210
08008220
08008230
08008240
08008250
08008260
08008270
08008280
08008290
08008300
08008310
08008320
08008330
08008340
08008350
08008360
08008370
08008380
08008390
08008400
08008410
08008420
08008430
08013910
08013912
08013914
08013916
08013918
08013920
08013922
08013923
08013924
08013926
08013940
08013942
08014000
08014010
08014020
08014040
08014050
08014060
08014062

```

```

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]x100+Y.[36:6]x10+Y.[42:6];
YI:=Y.[18:6]x10+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 DWT ELSE
BEGIN D:=D-H; M:=M+1
END;
DWT:
H:=TIME1 DIV 216000;
MIN:=(TIME1 DIV 3600) MOD 60;
IF M LSS 2 THEN
BEGIN Q:=M+1; P:=Y-1
END ELSE
BEGIN Q:=M-1; P:=Y
END;
M:=M+1;
FORM(B,TIME1:=((Qx26-2)DIV 10+D+P+P.[36:10]+1)MOD 7,
M,D,Y,Q:=H MOD 12, Q:=MIN MOD 10+(MIN DIV 10)x64,
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 J=ITEM#;
PROCEDURE GETALINE(C,S,L,R,RD,D,LEN);
VALUE LEN; INTEGER C,S,L,LEN;
ARRAY R[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,R);
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 CHRS 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

```

```

08014064
08014066
08014068
08014070
08014072
08014074
08014076
08014078
08014080
08014082
08014084
08014086
08014088
08014090
08014092
08014100
08014110
08014120
08014130
08014140
08014150
08014160
08014170
08014180
08014190
08014200
08014210
08014220
08014230
08014240
08014250
08014260
08014270
08014280
08014282
08014284
08014286
08014288
08014290
08014300
08014310
08014320
08014325
08014327
08014329
08014331
08014333
08014335
08014337
08014339
08014341
08014343
08014345
08014347
08014349
08014351
08014355
08014359
08014363
08014367
08014369
08014371
08014375
08014379
08014383
08014387
08014391
08014395
08014399
08014403
08014407
08014411
08014415
08014419
08014423
08014427
08014431
08014435
08014439
08014443
08014447
08014451
08014455

```

```

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 FILLING 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;
          LI:=1;
        WHILE L NEQ 0 DO

```

```

08014459
08014463
08014467
08014471
08014475
08014479
08014483
08014487
08014491
08014495
08014499
08014503
08014507
08014511
08014515
08014519
08014523
08014527
08014529
08014530
08014531
08014535
08014539
08014540
08014543
08014547
08014551
08014555
08014559
08014565
08014567
08014569
08014570
08014571
08014575
08014579
08014583
08014587
08014591
08014595
08014597
08014599
08014603
08014607
08014619
08014620
08014639
08014643
08014647
08014649
08014650
08014651
08014655
08014659
08014663
08014667
08014671
08014675
08014679
08014683
08014687
08014691
08014695
08014699
08014703
08014707
08014711
08014712
08014713
08014714
08014715
08014716
08014717
08014718
08014719
08014720
08014721
08014723
08014727
08014731
08014735
08014739
08014743

```

```

      BEGIN
      GETLINE(C,S,L,B,RD,D,LEN);
      GT1:=STORESEQ(CS, BUFFER, L);
      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);
      GETLINE(C,S,L,B,RD,D,LEN);
      T[1].DIMPTR:=STORESEQ(WS, BUFFER, L);
      * PUTS DIMVECT INTO WORKSPACE
      GETLINE(C,S,L,B,RD,D,LEN); * # BLOCKS
      SIZE:=L-1;
      FOR K:=0 STEP 2 UNTIL SIZE DO
      BEGIN
      GETLINE(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(1, 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 0 THEN GO SKIP;
      IF NDIR:=D[0] NEQ 0 THEN
      BEGIN N:=L*BSPPACES+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");
      FOR:=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;
ARRAY T[0:1];
J:=SIZE(V:=VARIABLES)-1;
FOR I:=0 STEP 1 UNTIL J DO
  BEGIN K:=CONTENTS(V,I,T);
  IF GETFIELD(T,7,1)=FUNCTION THEN
    FOR L:=EPTF,FSQF DO % GET RID OF STORAGE
      IF G:=GETFIELD(T,L,FFL) NEQ 0 THEN RELEASEUNIT(G);
    END;
  RELEASEUNIT(V);
  VARIABLES:=0; VARSIZE:=0;
  CURRENTMODE:=0; J:=SIZE(WS)-1;
  FOR I:=1 STEP 1 UNTIL J DO DELETE1(WS,I);
  STOREPSR;
END;
PROCEDURE ELIMWORKSPACE(WS); VALUE WS; INTEGER WS;
BEGIN LABEL QQQ; QQQ;
IF WORKSPACE NEQ 0 THEN
  BEGIN
    PURGEWORKSPACE(WS); RELEASEUNIT(WS);
  %
  END ELSE SPOUT(8015222);
END;
PROCEDURE SAVEWORKSPACE(NAME1,NAME2,LOCKFILE);
VALUE NAME1,NAME2,LOCKFILE;
REAL NAME1,NAME2,LOCKFILE;
BEGIN
  SAVE FILE DISK DISK (NAREAS,SIZEAREAS)
  (2,WDSPPERREC,WDSPERBLK,SAVE 100);
  INTEGER PROCEDURE WR(D,N,A); VALUE N; INTEGER N;
  FILE D; ARRAY A[0];
  BEGIN REAL STREAM PROCEDURE CON(A); VALUE A;
    BEGIN SI:=LOC A; DI:=LOC CON; DS:=8DEC END;
    STREAM PROCEDURE CLEANER(A);
      BEGIN DI:=A; WDSPPERREC(DS:=8LIT".") END;
    A[WDSPPERREC-1]:=CON(N);
    WRITE(D[N],WDSPPERREC,A[*]);
    WR:=N+1; CLEANER(A);
  END;
  PROCEDURE PUTAWAY(C,J,WR,D,N,M,B,L); VALUE L,J;
  INTEGER L,C,J,N,M;
  ARRAY B[0]; INTEGER PROCEDURE WR; FILE D;
  BEGIN INTEGER P,K;
  IF C+2 GTR L THEN
    BEGIN TRANSFER(J,6,B,C,1); N:=WR(D,N,B); C:=1;
      TRANSFER(J,7,B,0,1);
    END ELSE
    BEGIN TRANSFER(J,6,B,C,2); C:=C+2
      END;
  WHILE J NEQ 0 DO
    IF J GTR K:=(L-C) THEN
      BEGIN TRANSFER(BUFFER,P,B,C,K);
        N:=WR(D,N,B); J:=J-K; C:=0; P:=P+K
      END ELSE
      BEGIN TRANSFER(BUFFER,P,B,C,J); C:=C+J; J:=0
        END;
    IF C=L THEN
      BEGIN N:=WR(D,N,B); C:=0
        END;
  END;
  PROCEDURE MOVETWO(U,B,M,WR,L,D);
  ARRAY U,B[0]; INTEGER M,L; INTEGER PROCEDURE WR; FILE D;
  BEGIN
  COMMENT PUTS 2 WORDS OF U IN B AND WRITES ON D IF A FULLRECORD;
  TRANSFER(U,0,B,M*8,16); % CONTENTS OF U INTO B
  M:=M+2;
  IF M GEQ WDSPPERREC-1 THEN % FULL RECORD
    BEGIN
      L:=WR(D,L,B);
      IF M GEQ WDSPPERREC THEN % 1 OVER FULL RECORD
        BEGIN
          TRANSFER(U,8,B,0,8);
          M:=1;
          END
        ELSE M:=0;
      END;
  END OF MOVETWO;
  INTEGER H,Q,M,N,I,L,S,J,K,LINE,MAX,PT,SO,C,LEN,W;
  REAL LSD,STP;
  LABEL SKIP;
  ARRAY T,U[0:1],DIR,B,SEX[0:WDSPPERREC];

```

```

08015502
08015503
08015504
08015505
08015506
08015507
08015508
08015509
08015510
08015511
08015512
08015513
08015514
08015515
08015516
08015517
08015518
08015519
08015520
08015521
08015522
08015523
08015524
08015525
08015526
08015527
08015528
08015529
08015530
08015531
08015532
08015533
08015534
08015535
08015536
08015537
08015538
08015539
08015540
08015541
08015542
08015543
08015544
08015545
08015546
08015547
08015548
08015549
08015550
08015551
08015552
08015553
08015554
08015555
08015556
08015557
08015558
08015559
08015560
08015561
08015562
08015563
08015564
08015565
08015566
08015567
08015568
08015569
08015570
08015571
08015572
08015573
08015574
08015575
08015576
08015577
08015578
08015579
08015580
08015581
08015582
08015583
08015584
08015585
08015586
08015587
08015588
08015589
08015590
08015591
08015592
08015593
08015594
08015595
08015596
08015597
08015598
08015599
08015600
08015601
08015602
08015603
08015604
08015605
08015606
08015607
08015608
08015609
08015610
08015611
08015612
08015613
08015614
08015615
08015616
08015617
08015618
08015619
08015620
08015621
08015622
08015623
08015624
08015625
08015626
08015627
08015628
08015629
08015630
08015631
08015632
08015633
08015634
08015635
08015636
08015637
08015638
08015639
08015640
08015641
08015642
08015643
08015644
08015645
08015646
08015647
08015648
08015649
08015650
08015651
08015652
08015653
08015654
08015655
08015656
08015657
08015658
08015659
08015660
08015661
08015662
08015663
08015664
08015665
08015666
08015667
08015668
08015669
08015670
08015671
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;
DIRIOJ:=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,FPT,FFL); % FUNCTION POINTER FIELD
    SQ:=GETFIELD(T,FSQ,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];
          UL1:=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);

```

```

08015567;
08015568;
08015569;
08015570;
08015571;
08015572;
08015573;
08015574;
08015575;
08015576;
08015577;
08015578;
08015579;
08015580;
08015581;
08015582;
08015583;
08015584;
08015585;
08015586;
08015587;
08015588;
08015589;
08015590;
08015591;
08015592;
08015593;
08015594;
08015595;
08015596;
08015597;
08015598;
08015599;
08015600;
08015601;
08015602;
08015603;
08015604;
08015605;
08015606;
08015607;
08015608;
08015609;
08015610;
08015611;
08015612;
08015613;
08015614;
08015615;
08015616;
08015617;
08015618;
08015619;
08015620;
08015621;
08015622;
08015623;
08015624;
08015625;
08015626;
08015627;
08015628;
08015629;
08015630;
08015631;
08015632;
08015633;
08015634;
08015635;
08015636;
08015637;
08015638;
08015639;
08015640;
08015641;
08015642;
08015643;
08015644;
08015645;
08015646;
08015647;
08015648;
08015649;
08015650;
08015651;
08015652;
08015653;
08015654;
08015655;
08015656;
08015657;
08015658;
08015659;
08015660;
08015661;
08015662;
08015663;
08015664;
08015665;
08015666;
08015667;
08015668;
08015669;
08015670;
08015671;
08015672;
08015673;
08015674;
08015675;
08015676;
08015677;
08015678;
08015679;
08015680;
08015681;
08015682;
08015683;
08015684;
08015685;
08015686;
08015687;
08015688;
08015689;
08015690;
08015691;
08015692;
08015693;
08015694;
08015695;
08015696;
08015697;
08015698;
08015699;
08015700;
08015701;
08015702;
08015703;
08015704;
08015705;
08015706;
08015707;
08015708;
08015709;
08015710;
08015711;
08015712;
08015713;
08015714;
08015715;
08015716;
08015717;
08015718;
08015719;
08015720;
08015721;
08015722;
08015723;
08015724;
08015725;
08015726;
08015727;
08015728;
08015729;
08015730;
08015731;
08015732;
08015733;
08015734;
08015735;
08015736;
08015737;
08015738;
08015739;
08015740;
08015741;
08015742;
08015743;
08015744;
08015745;
08015746;
08015747;
08015748;
08015749;
08015750;
08015751;
08015752;
08015753;
08015754;
08015755;
08015756;
08015757;
08015758;
08015759;
08015760;
08015761;
08015762;
08015763;
08015764;
08015765;
08015766;
08015767;
08015768;
08015769;
08015770;
08015771;
08015772;
08015773;
08015774;
08015775;
08015776;
08015777;
08015778;
08015779;
08015780;
08015781;
08015782;
08015783;
08015784;
08015785;
08015786;
08015787;
08015788;
08015789;
08015790;
08015791;
08015792;
08015793;
08015794;
08015795;
08015796;
08015797;
08015798;
08015799;
08015800;
08015801;
08015802;
08015803;
08015804;
08015805;
08015806;
08015807;
08015808;
08015809;
08015810;
08015811;
08015812;
08015813;
08015814;
08015815;
08015816;
08015817;
08015818;
08015819;
08015820;
08015821;
08015822;
08015823;
08015824;
08015825;
08015826;
08015827;
08015828;
08015829;
08015830;
08015831;
08015832;
08015833;
08015834;
08015835;
08015836;
08015837;
08015838;
08015839;
08015840;
08015841;
08015842;
08015843;
08015844;
08015845;
08015846;
08015847;
08015848;
08015849;
08015850;
08015851;
08015852;
08015853;
08015854;
08015855;
08015856;
08015857;
08015858;
08015859;
08015860;
08015861;
08015862;
08015863;
08015864;
08015865;
08015866;
08015867;
08015868;
08015869;
08015870;
08015871;
08015872;
08015873;
08015874;
08015875;
08015876;
08015877;
08015878;
08015879;
08015880;
08015881;
08015882;
08015883;
08015884;
08015885;
08015886;
08015887;
08015888;
08015889;
08015890;
08015891;
08015892;
08015893;
08015894;
08015895;
08015896;
08015897;
08015898;
08015899;
08015900;
08015901;
08015902;
08015903;
08015904;
08015905;
08015906;
08015907;
08015908;
08015909;
08015910;
08015911;
08015912;
08015913;
08015914;
08015915;
08015916;
08015917;
08015918;
08015919;
08015920;
08015921;
08015922;
08015923;
08015924;
08015925;
08015926;
08015927;
08015928;
08015929;
08015930;
08015931;
08015932;
08015933;
08015934;
08015935;
08015936;
08015937;
08015938;
08015939;
08015940;
08015941;
08015942;
08015943;
08015944;
08015945;
08015946;
08015947;
08015948;
08015949;
08015950;
08015951;
08015952;
08015953;
08015954;
08015955;
08015956;
08015957;
08015958;
08015959;
08015960;
08015961;
08015962;
08015963;
08015964;
08015965;
08015966;
08015967;
08015968;
08015969;
08015970;
08015971;
08015972;
08015973;
08015974;
08015975;
08015976;
08015977;
08015978;
08015979;
08015980;
08015981;
08015982;
08015983;
08015984;
08015985;
08015986;
08015987;
08015988;
08015989;
08015990;
08015991;
08015992;
08015993;
08015994;
08015995;
08015996;
08015997;
08015998;
08015999;

```



```

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,1,7)) NEQ 0 THEN
BEGIN GTAL0]←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

```

```

08015888
08015889
08015891
08015892
08015894
08015900
08015920
08015922
08015930
08015940
08015950
08015952
08015954
08015956
08015958
08015959
08015960
08015961
08015962
08015963
08015964
08015966
08015992
08016000
08016005
08016008
08016009
08016010
08016011
08016012
08016014
08016016
08016018
08016020
08016022
08016024
08016025
08016026
08016028
08016030
08016032
08016033
08016034
08016035
08016036
08016038
08016040
08016042
08016044
08016046
08016048
08016050
08016052
08016054
08016060
08016070
08016080
08016090
08016100
08016110
08016120
08016125
08016130
08016131
08016132
08016133
08016134
08016135
08016138
08016140
08016150
08016160
08016165
08016170
08016180
08016190
08016200
08016205
08016210
08016220
08016230
08016240
08016250

```

```

-----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;
REAL PROCEDURE LINENUMBER(R); VALUE R; REAL R;
BEGIN
  REAL STREAM PROCEDURE CON(R); VALUE R;
  BEGIN SI:=LOC R; DI:=LOC CON; DS:=BDEC
  END;
  LINENUMBER:=CON( ENTIER( (R+.00005)x10000))
END;
DEFINE DELIM="###", ENDCHR="s##";
BOOLEAN PROCEDURE WITHINLINE(COMMAND,OLD,NEW,CHAR,WORD);
VALUE COMMAND,CHAR,WORD; INTEGER COMMAND,CHAR,WORD;
ARRAY OLD,NEW(0); BEGIN
BOOLEAN STREAM PROCEDURE WITHINLINE(COMMAND,OLD,NEW,CHAR,WORD);
VALUE COMMAND,CHAR,WORD;
BEGIN
  LOCAL OLDLINE,NEWLINE,F,BCHR;
  LOCAL N,M,T;

```

08017970
08018000
08018010
08018012
08018020
08018022
08018023
08018030
08018100
08018110
08018120
08018130
08018140
08018150
08018160
08018170
08018180
08018190
08018200
08018210
08018215
08018220
08018225
08018230
08018235
08018240
08018241
08018242
08018243
08018245
08018250
08018255
08018260
08018265
08018270
08018275
08018300
08018305
08018310
08018315
08018320
08018330
08018462
08018470
08018577
08018580
08018767
08018770
08018780
08018930
08018942
08018965
08018970
08018975
08018980
08018985
08018990
08018995
08018999
08019000
08019010
08019020
08019100
08019200
08019210
08019300
08019400
08030000
08030010
08030020
08030030
08030040
08030050
08030060
08030080
08030082
08030084
08030086
08030100
08030102
08030110
08030120
08030130


```

OWN BOOLEAN EDITMODE;
DEFINE FPT=FUNCPOINTER#,
FSQ=FUNCSEQ#,
SEQ=CURLINE#,
INC=INCREMENT#,
MODE=SPECMODF#,
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(XEQUATE);
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);

```

```

09003000
0900400
0900410
0900420
09004300
0900431
0900440
0900500
09005100
09005110
0900512
0900513
0900514
09005150
0900516
0900517
0900518
09005200
09005300
0900540
0900550
09005600
09005700
0900600
0900700
0900800
09009000
09010000
0901100
0901200
09013000
09014000
0901500
0901600
0901700
09018000
09019000
0901910
0901920
09019300
09019400
0902000
0902100
0902200
09023000
0902400
0902500
0902600
09027000
09028000
0902900
0903000
09031000
09032000
0903300
0903400
0903500
09036000
09036100
0903700
0903800
09039000
09040000
0904100
0904200
0904300
09044000
09045000
0904600
0904700
09048000
09048500
0904900
0905000
0905100
09052000
09052100
0905220
0905230
09052400
09052500
0905260
0905270

```

```

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
EDITORIVER(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);
INITRUFF(BUFFER,BUFFSIZE);
K:=CONTENTS(SQ,L,BUFFER);
RESCANLINE;

```

```

0905281
0905301
0905401
0905501
0905601
0905701
0905801
0905810
0905900
0906000
0906100
0906200
0906300
0906400
0906400
0906500
0906600
0906700
0906800
0906810
0906820
0906830
0906840
0906850
0906900
0906910
0906911
0906911
0906912
0906920
0906930
0906940
0907000
0907100
0907200
0907210
0907220
0907230
0907240
0907250
0907260
0908100
0908200
0908300
0908400
0908500
0908600
0908700
0908800
0908900
0909000
0909100
0909200
0909300
0909400
0909500
0909600
0909700
0909800
0909900
0910000
0910100
0910200
0910300
0910400
0910500
0910600
0910700
0910800
0910900
0911000
0911100
0911200
0911300
0911400
0911500
0911600
0911700
0911800
0911900
0912000
0912100
0912200
0912300

```



```

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 ENHANDLER;
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,FPT,FFL);
  FUNCSEQ:=GETFIELD(GTA,FSQ,FFL);
  GT3:=CURLINE:=TOPLINE(FPT);
  CHECKSEQ(CURLINE,GT3,INC); %SET THE INCREMENT
  COMMENT THE CURRENTLINE IS SET TO THE LAST LINE OF THE
  FUNCTION;
  FUNCSEQ:=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., OAAAAAAA
  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 OXXXXXX;
  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,
  HEADERCTADD.[1:23],TADD.[24:24],OLDBUFFER));
  SETFIELD(GTA,0,8,0);
  STOREORD(F:=FUNCPOINTER:=NEXTUNIT,GTA,0);

```

```

09198270
09198280
09198282
09198290
09199000
09200000
09201000
09208000
09209000
09210000
09211000
09212000
09213000
09214000
09215000
09216000
09217000
09218000
09219000
09220000
09221000
09222000
09223000
09224000
09224100
09224200
09224300
09225000
09225100
09225110
09225115
09225200
09225300
09225310
09225400
09225500
09225600
09225700
09225800
09225900
09226000
09226100
09226200
09226300
09226400
09226500
09226600
09226700
09226800
09226900
09226910
09226920
09226930
09227000
09227100
09227200
09227300
09227400
09227410
09227500
09227510
09227520
09227522
09227530
09227534
09227538
09227542
09227546
09227550
09227554
09227558
09227562
09227566
09227570
09227580
09227584
09227558
09227522
09227600
09227700
09227800
09227900
09228000

```

```

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;
  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;
  END;
GTA[1]:=0&ENTIER(A[1])[RETURN]&ENTIER(A[2])[NUMBERARGS];
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 % VARIABLES=0, MAKE UP A DIRECTORY
  BEGIN
    GT3:=0; GT2:=1; GO TO HEADERSTORE
  END
ELSE % HEADER SYNTAX IS BAD
  GO TO ENHANDLER;
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,FPT,FEL,FUNCTION);
    SETFIELD(GTA,FSQ,FFL,FUNCTION);
    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
    FORGETTIFELLA; 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
        INITBUFFER(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
        END
      ELSE MODE:=0;
    LOWER:=L+1;
    IF LOWER GTR UPPER THEN
      BEGIN
        IF MODE=DISPLAYING THEN

```

```

092228004
092228006
092228008
092228010
092228012
092228014
092228016
092228018
092228019
092228020
092228022
092228024
092228030
092228040
092228050
092228052
092228060
092228070
092228080
092228100
092228200
092228202
092228210
092228220
092228230
092228300
092228400
092228500
092228600
092228700
092228800
092228900
092229000
092229100
092229200
092229300
092229400
092229500
092229600
092229700
092229800
092229900
092300000
092300010
092300100
092300200
092300300
092300305
092300310
092300400
092300500
092300600
092300700
092300800
092300900
092310000
092310100
092310200
092310300
092311000
092311020
092311040
092311060
092311080
092311100
092311110
092311114
092311118
092311122
092311124
092311126
092311130
092311134
092311138
092311142
092311146
092311150
092311154
092311158
092311162
092311166
092311170
092311200
092311300

```

```

FORMWD(3,"1      ");
MODE:=0;
END;
GO TO ENDHANDLER
END;
END ; %OF BLOCK STARTED ON LINE 9225115 //////////////////////////////////

```

```

IF ERR=0 AND EOB=0 THEN
SHORTCUT: BEGIN LABEL RGTBRACK,DELOPTION; %////////////////////////////////////
IF DELV THEN FINISHUP ELSE
IF LFTBRACKET THEN
BEGIN
IF SCAN THEN
IF ROUND(FPT) THEN
BEGIN L:=ACCUM[0];
IF SCAN THEN
IF QUADV OR EDITMODE:=(QUOTEQUAD) THEN
IF SCAN THEN
IF ROUND(FPT) THEN
BEGIN U:=ACCUM[0];
RGTBRACK:
IF SCAN AND RGTBRACKET THEN
IF(IF EDITMODE THEN FALSE ELSE SCAN) THEN
IF DELV THEN
BEGIN ERR:=DISPLAY(L,U,FPT,FSQ);
DELTOG:=1;
END
ELSE ERR:=1
ELSE ERR:=DISPLAY(L,U,FPT,FSQ)
ELSE ERR:=2
END
ELSE
IF RGTBRACKET THEN
IF(IF EDITMODE THEN FALSE ELSE SCAN) THEN
IF DELV THEN
BEGIN ERR:=DISPLAY(L,L,FPT,FSQ);
DELTOG:=1;
END
ELSE ERR:=3
ELSE ERR:=DISPLAY(L,L,FPT,FSQ)
ELSE ERR:=4
ELSE ERR:=5
ELSE
IF RGTBRACKET THEN
BEGIN TADD:=ADDRESS;
IF SCAN THEN
IF IDENT AND ACCUM[0]="6DELETE" THEN
IF SCAN THEN
IF LFTBRACKET THEN
DELOPTION:
IF SCAN AND BOUND(FPT) THEN
BEGIN U:=ACCUM[0];
IF SCAN AND RGTBRACKET THEN
IF SCAN THEN
IF DELV THEN
BEGIN ERR:=DELETE(L,U,FPT,FSQ);
FINISHUP
END
ELSE ERR:=6
ELSE ERR:=DELETE(L,U,FPT,FSQ)
END
ELSE ERR:=8
ELSE
IF DELV THEN
BEGIN ERR:=DELETE(L,L,FPT,FSQ);
FINISHUP
END
ELSE ERR:=9
ELSE ERR:=DELETE(L,L,FPT,FSQ)
ELSE
IF LFTBRACKET THEN GO TO DELOPTION ELSE
BEGIN CHECKSEQ(SEQ,L,INC);
CLEANBUFFER(BUFFER,BUFFSIZE,TADD);
ADDRESS:=ABSADDR(BUFFER); ITEMCOUNT:=0;
IF SCAN THEN GO TO SHORTCUT
END
ELSE ERR:=DELETE(L,L,FPT,FSQ)
END
ELSE ERR:=10

```

```

09231400
09231500
09231600
09231700
09231800
09232000
09233000
09234000
09235000
09236000
09237000
09238000
09239000
09240000
09241000
09242000
09243000
09244000
09245000
09246000
09247000
09248000
09249000
09250000
09251000
09252000
09253000
09254000
09255000
09256000
09257000
09258000
09259000
09260000
09261000
09262000
09263000
09264000
09265000
09266000
09267000
09268000
09269000
09270000
09271000
09272000
09273000
09274000
09275000
09276000
09277000
09278000
09279000
09280000
09281000
09282000
09283000
09284000
09285000
09286000
09287000
09288000
09289000
09290000
09291000
09292000
09293000
09294000
09295000
09296000
09297000
09298000
09299000
09300000
09301000
09302000
09303000
09304000
09305000
09306000
09307000
09308000
09309000

```

```

ELSE ERR:=11
END ELSE
IF QUADV OR EDITMODE:=(QUOTEQUAD) THEN
  BEGIN L:=0; U:=9999.9999; GO TO RGTBRACK
END ELSE
IF IOTA THEN
  IF SCAN AND RGTBRACKET AND FPT NEQ 0 THEN
    BEGIN IF SCAN THEN
      IF DELV THEN DELTOG:=1 ELSE ERR:=15;
      IF ERR = 0 THEN
        BEGIN MODE:=RFSEQUENCING; CURLINE:=INCREMENT:=1;
        SETFIELD(GTA,0,8,0);
        GT1:=SEARCHORD(FPT,GTA,GT2,8);
        LOWER:=GT2+1; UPPER:=FUNCSIZE-1
        END
      END
    ELSE ERR:=14
    ELSE ERR:=12
    ELSE ERR:=13
  END
ELSE
  IF CURLINE=0 THEN %CHANGING HEADER
  ERR:=26 ELSE
  IF ERR:=OLDLABCONFLICT(FPT,LINENUMBER(SEQ))=0 THEN
    BEGIN
    IF NOSYNTAX=0 THEN PROCESS(XEQUTE);
    IF ERR:=STOREAWAY(FPT,FSQ,BUFFER,SEQ)=0 THEN SEQ:=SEQ+INC;
    END;
    IF ERR NEQ 0 THEN
    BEGIN FORMWD(2,"5ERROR ");
    NUMBERCON(ERR,ACCUM); ERR:=0;
    EOB:=1;
    FORMROW(1,1,ACCUM,2,ACCUM[0],[1:11]);
    END;
  END; %OF BLOCK STARTED ON LINE 9238000 //////////////////////////////////////
  ENHANDLER;
  IF BOOLEAN(SUSPENSION) THEN BEGIN
    FILL ACCUM[*] WITH "ABORT SU", "SP, FNS.";
    FORMROW(3,0,ACCUM,0,16); INDENT(0); TERPRINT;
  END ELSE
  IF MODE=0 THEN
  BEGIN
  IF BOOLEAN(DELTG) THEN FINISHUP;
  INDENT(-CURLINE); TERPRINT;
  END;

END;
EXPOVR:=FAULTL; INTOVR:=FAULTL; INDEXF:=FAULTL;
FLAG:=FAULTL; ZERO:=FAULTL;
INITIALIZETABLE;
TRYAGAIN;
IF FALSE THEN %ENTERS WITH A FAULT.
FAULTL;
  BEGIN SPOUT(09334300); %SEND A MESSAGE TO SPO
  BEGIN CSTATION,APLOGGED:=0; CSTATION.APLHEADING:=0
  END
END;
APLMONITOR;
ENDOFJOB;
FINIS;
WRAPUP;

END;
END;END.

```

```

09310000
09311000
09312000
09313000
09314000
09314200
09314300
09314310
09314330
09314340
09314350
09314400
09314410
09314420
09314500
09314600
09314700
09315000
09316000
09317000
09318000
09318100
09318110
09319000
09320000
09321000
09322000
09323000
09324000
09325000
09326000
09327000
09328000
09329000
09330000
09330100
09330102
09330104
09330106
09330108
09330110
09330112
09330120
09330200
09330210
09331000
09332000
09332100
09332200
09333000
09334000
09334100
09334200
09334300
09334400
09334500
09334600
09334700
09335000
09336000
09337000
09338000
09339000
09340000
09341000
99999999

```

TOTAL LOGICAL RECORDS= 7273
 END OF JOB.