A Source Listing

of

H O P E

A Human Oriented Program Editor

Management:
Henry Ledgard

Design:
Andrew Singer
Jon Hueras

Implementation:
Daryl Winters

May 1978

This work was supported by the National Science Foundation.

This is a COINS Technical Report numbered 78-9, University of Massachusetts Amherst, 01003.

```
 1
 2 (*$E+,T-,P-*)
 3
 4 PROGRAM HOPE (TERMNAL, SCRATCH, COMMENT, OUTPUT);
 5
 6 (*************************************************************************)
 7 (*                                                                       *)
 8 (* TITLE:   HUMAN ORIENTED PROGRAM EDITOR (HOPE)                         *)
 9 (*                                                                       *)
10 (* AUTHORS: DARYL R. WINTERS                                             *)
11 (*          HENRY F. LEDGARD                                             *)
12 (*                                                                       *)
13 (*          COMPUTER AND INFORMATION SCIENCE                             *)
14 (*          UNIVERSITY OF MASSACHUSETTS, AMHERST                         *)
15 (*                                                                       *)
16 (* SUMMARY: THIS PROGRAM IS A PROTOTYPE IMPLEMENTATION OF                *)
17 (*          A HUMAN ORIENTED PROGRAM EDITOR. FOR A COMPLETE              *)
18 (*          DESCRIPTION OF THE USE OF HOPE REFER TO:                     *)
19 (*                                                                       *)
20 (*             A USER'S GUIDE TO HOPE                                    *)
21 (*             COINS TECHNICAL REPORT 78-05                              *)
22 (*             UNIVERSITY OF MASSACHUSETTS, AMHERST                      *)
23 (*                                                                       *)
24 (*                                                                       *)
25 (* FILE DESCRIPTIONS:                                                    *)
26 (*                                                                       *)
27 (*    TERMNAL: A FILE ASSOCIATED WITH A TIME-SHARING USER'S              *)
28 (*             TERMINAL FROM WHICH THE USER WILL ENTER                   *)
29 (*             REQUESTS AND TO WHICH HOPE WILL ISSUE MESSAGES.           *)
30 (*                                                                       *)
31 (*    SCRATCH: A FILE USED INTERNALLY BY HOPE FOR MANAGEMENT             *)
32 (*             OF THE USER'S PERMANENT FILES.                            *)
33 (*                                                                       *)
34 (*    COMMENT: A FILE WHICH WILL RECIEVE ALL OF THE USER'S               *)
35 (*             COMPLAINTS ISSUED DURING THE SESSION.                     *)
36 (*                                                                       *)
37 (*    OUTPUT:  THE STANDARD PASCAL FILE FOR RUNTIME MESSAGES.            *)
38 (*                                                                       *)
39 (*************************************************************************)
40
41 (*$Z *)
```

```
 42
 43 CONST
 44       REQPROMPT = '--':        (* REQUEST PROMPT *)
 45       ADDPROMPT = '++':        (* ADD CHARACTERS PROMPT *)
 46       CHKPROMPT = '//':        (* CAUTION CHECK PROMPT *)
 47       MSGPREFIX = '**':        (* MESSAGE PREFIX *)
 48       LSTPREFIX = '  ';        (* LISTING PREFIX *)
 49
 50       ERRORPOINTER = '/':      (* POINTS TO ERROR IN LINE *)
 51
 52       BLANK = ' ':
 53
 54       MAXSTRLEN = 140:         (* MAXIMUM TEXT STRING LENGTH *)
 55       MAXSYMLEN = 3:           (* MAXIMUM LENGTH OF SPECIAL SYMBOL *)
 56       MAXKEYLEN = 10:          (* MAXIMUM LENGTH (+1) OF KEYWORD *)
 57
 58       MAXFILENAMELEN = 7:      (* MAXIMUM FILENAME LENGTH *)
 59       MAXSTATENUM    = 6:      (* MAXIMUM STATE NUMBER IN PARSE *)
 60
 61       MAXNUMCHARS    = 50000: (* 40 PAGES OF 50 LINES  *)
 62
 63       ENDREQUESTSYM  = ';':   (* REQUEST SEPERATOR *)
 64
 65       NULLSTRLEN    = 0:       (* LENGTH OF NULL TEXT *)
 66       NULLPOINTER   = 0:       (* POINTER TO NULL SEQUENCE *)
 67       NULLREPEATER  = 0:       (* NULL VALUE FOR REPEATER *)
 68       NULLSYMLEN    = 0:       (* NULL SPECIAL SYMBOL LENGTH *)
 69
 70       OUTMARKER    = '@':      (* INTERNAL MARGIN MARKERS *)
 71       INMARKER     = '^':
 72
 73       DEFAULTRPTR = 1:         (* DEFAULT REPEATER *)
 74       REPEATALL    = MAXINT:   (* INTERNAL REPRESENTATION *)
 75
 76       INPUTFILE    = 'INPUT*     ':   (* INPUT DEVICE (TERMINAL) *)
 77       EMPTYFILE    = '           ':   (* UNKNOWN FILE *)
 78
 79 (* UNASS DEPENDANT CONTROL BYTES *)
 80
 81       TRANSPARENT     = ':E':  (* 0005B INITIATE TRANSPARENT INPUT *)
 82       NOLINEFEED      = ':K':  (* 0013B INHIBITS LINE FEED *)
 83       PRINTASCII      = ':I':  (* 0011B BEGINS ASCII OUTPUT *)
 84       SOUNDSIGNAL     = '^*':  (* 7647B SEND TONE SIGNAL *)
 85
 86       BREAKCODE       = '^':   (* 76B UPPER/LOWER CASE CONVERSION *)
 87       ESCAPECODE      = '@':   (* 74B *)
 88
 89 (* UNASS DEPENDANT CHARACTER CODES *)
 90
 91       ASCOUTMARKER = '@A':     (* ASCII DISPLAY MARGIN MARKERS *)
 92       ASCINMARKER  = '@B':
 93
 94       LINEMARK = '*':          (* SYSTEM ADDED CHARACTER *)
 95       DELCHAR  = '5':          (* 40B - ASCII DELETE *)
 96       BSCHAR   = '/':          (* 50B - ASCII BACKSPACE *)
 97       ESCCHAR  = 'A':          (* 01B - ASCII @ SYMBOL *)
 98       BRKCHAR  = 'B':          (* 02B - ASCII ^ SYMBOL *)
 99       CLNCHAR  = ':':          (* 04B - ASCII : SYMBOL *)
100       COLON    = ':':          (* 003 - ASCII TO NORMAL *)
101       LSTPRINT = '3':          (* 36B - LAST PRINTABLE ASCII *)
102
103
104 (*$Z *)
```

```
105
106 TYPE
107       KEYSYMBOL = (FORWARD,     BACKWARD,    LIST,        DELETE,
108                    TRANSFER,    COPY,        MAKE,        INSERT,
109                    ASSUME,      SHOW,        PRESERVE,    DESTROY,
110                    UNDO,        QUIT,        AFTER,       BEFORE,
111                    OVER,        FILESYM,     MARGIN,      ELLIPSIS,
112                    JOKER,       TOP,         BOTTOM,      ALPHABET,
113                    CATALOG,     ALL,         HERE,        NULL,
114                    ASCII,       NORMAL,      ISSYM,       INTO,
115                    FIRST,       LAST,        YES,         NO,
116                    GRIPE,       UNKNOWN):
117
118
119       ERROPMSG = (ERRISSYM,          ERRFILENAME,
120                   ERRSPECIALSYM,     ERRNUMBER,
121                   ERRFIRST,          ERRLAST,
122                   ERRNORMAL,         ERRINTO,
123                   ERRENDREQUEST,     ERRNOREQUEST,
124                   ERROPTION,         ERRMARGINSYM,
125                   ERRELLIPSISSYM,    ERRJOKERSYM,
126                   ERRELLPOSITION,    ERRASCIIJOKER,
127                   ERRCANTSHOW,
128                   ERRUNDO,           ERRNOTFOUND,
129                   ERRNOCURFILE,      ERRNOTALLLINES,
130                   ERRCANTSAVE,       ERRNOROOM,
131                   ERRMISSINGTEXT,    ERRTEXTLENGTH,
132                   ERRCHARACTERS,     ERRSTOPREQUEST);
133
134
135       VALUESYM = (VALFILENAME,   VALMARGINSYM,
136                   VALJOKERSYM,   VALELLIPSIS,
137                   VALALPHABET,   VALTOP,
138                   VALCURLINE,    VALBOTTOM):
139
140
141       INFOMSG = (INTRO,          FINAL,
142                  UNDONE,         NUMLINES,
143                  CANTUNDO,       NOFILEFOUND,
144                  ONLYNOYES,      NOTDONE,
145                  REENTER,        LONGLINE,
146                  NEWFILE,        RESTRICTION,
147                  ENTER,          FILEINSERT,
148                  ENDTEXT,        CATLIST,
149                  FILEHEAD,       EMPTYCAT,
150                  ENTERGRIPE,     GRIPESAVED):
151
152
153       CHECKPROBLEM = (OVERWRITEFILE,       FORGETFILE,
154                       CHKSINGLECHAR,       CHKDOUBLECHAR,
155                       DOREQUEST,           OKTOQUIT,
156                       ELIMINATEFILE):
157
158
159       FILETYPE = (NEW,  OLD):
160
161       DIRECTION = (UP,  DOWN):
162
163
164
165       STRING     = PACKED ARRAY [1..MAXSTRLEN] OF CHAR;
166
167       STRINGINDEX = 0..MAXSTRLEN;
```

```
168
169
170       SYMSTRING = PACKED ARRAY [1..MAXSYMLEN] OF CHAR;
171
172       SYMINDEX = 0..MAXSYMLEN;
173
174       SYMINFO = RECORD
175                    VAL : SYMSTRING;
176                    LEN : SYMINDEX;
177                 END;
178
179
180       DOUBLECHAR = PACKED ARRAY [1..2] OF CHAR;
181
182       ALFA = PACKED ARRAY [1..10] OF CHAR; (* CDC WORD *)
183
184
185       REQUESTINFO = RECORD
186                        NAME        : KEYSYMBOL;
187                        REPEATER    : INTEGER;
188                        TEXTSTR,
189                        NEWTEXTSTR  : STRING;
190                        TEXTLEN,
191                        NEWTEXTLEN  : STRINGINDEX;
192                        PLACE,
193                        OPTION,
194                        VALUE       : KEYSYMBOL;
195                        SPECIALSYM  : SYMINFO;
196                        FILENAME    : ALFA;
197                     END;
198
199
200       KEYSTRING = PACKED ARRAY [1..MAXKEYLEN] OF CHAR;
201
202       KEYTABLE = ARRAY [KEYSYMBOL] OF KEYSTRING;
203
204
205       FINITESTATE = 1..MAXSTATENUM;
206
207
208       INTERNALFILE = PACKED ARRAY [1..MAXNUMCHARS] OF CHAR;
209
210       INCOREINDEX = 0..MAXINT;
211
212       INCOREPTRS = RECORD
213                       FIRST,
214                       LAST : INCOREINDEX;
215                    END;
216
217
218       CHRSYMBOLS = (WILDCARD, FREESTRING, CONSTANT);
219
220       TOKENS = ARRAY [1..MAXSTRLEN] OF CHRSYMBOLS;
221
222       POINTERS = ARRAY [1..MAXSTRLEN] OF INCOREPTRS;
223
224       TOKENINFO = RECORD
225                      SYM : TOKENS;
226                      CHR,
227                      JOK : STRING;
228                      ELL : POINTERS;
229                   END;
230
231 (*$Z *)
```

```
232
233 VAR
234
235 (* CONSTANT STRUCTURES *)
236
237     NULLTEXT    : STRING:
238     NULLSYM     : SYMINFO:
239     MAXSEQ,
240     NULLSEQ     : INCOREPTRS:
241     NULLREQUEST: REQUESTINFO:
242     NULLPATTERN: TOKENINFO:
243
244     LM,                                  (* LEFT MARGIN SYMBOL *)
245     RM          : DOUBLECHAR:            (* RIGHT MARGIN SYMBOL *)
246
247     KEYWORD     : KEYTABLE:              (* KEYWORD REPRESENTATIONS *)
248
249
250 (* PARSER GLOBAL VARIABLES *)
251
252     NEXTCHAR    : CHAR:                  (* NEXT CHARACTER (FOR PARSER) *)
253
254     TEXTLINE    : STRING:                (* REQUEST INPUT LINE *)
255
256     CURLINEPOS,                          (* CURRENT LINE POSITION *)
257     LINELENGTH  : STRINGINDEX:           (* INPUT LINE LENGTH *)
258
259     ENDOFLINE   : BOOLEAN:               (* END OF INPUT LINE *)
260
261     CHECKLEFT,
262     CHECKRIGHT  : BOOLEAN:               (* CONTROLS TOKENIZING OF TEXT *)
263
264
265 (* CURRENT REQUEST GLOBAL VARIABLES *)
266
267     DONE,                                (* TERMINATES HOPE SESSION *)
268     ERROR       : BOOLEAN:               (* SYNTACTIC OR SEMANTIC ERROR *)
269
270     ANSWER      : KEYSYMBOL:             (* ANSWER FROM SECURITY CHECK *)
271
272     VALIDINFO,                           (* CONTROLS INFO ON SCRATCH *)
273     SHOWLINE,                            (* SHOWS 'CL' MESSAGE *)
274     SHOWINPUT   : BOOLEAN:               (* LIST REQUEST AGAIN *)
275
276     REQUEST,                             (* CURRENT HOPE REQUEST *)
277     LASTREQUEST: REQUESTINFO:            (* LAST REQUEST PERFORMED *)
278
279     PATLEN,                              (* PATTERN LENGTH *)
280     REPLEN  : STRINGINDEX:               (* REPLACEMENT LENGTH *)
281
282     PATTERN,                             (* TOKENIZED PATTERN *)
283     REPLACEMENT : TOKENINFO:             (* REPLACEMENT PHRASE *)
284
285     LINESFOUND,                          (* NUMBER OF LINES IN MATCHES *)
286     MATCHNUM    : INTEGER:               (* NUMBER OF MATCHES TO FIND *)
287
288     TERMNAL,                             (* TERMINAL INPUT FILE *)
289     SCRATCH,                             (* GENERAL PURPOSE FILE *)
290     COMMENT : TEXT:                      (* USER GRIPES FILE *)
291
292 (*$Z *)
```

```
293
294 (* HOPE STATUS GLOBAL VARIABLES *)
295
296     MARGINSYM,
297     JOKERSYM,
298     ELLIPSISSYM,
299
300     TEMPMARGINSYM,
301     TEMPJOKERSYM,
302     TEMPELLIPSISSYM : SYMINFO:      (* EMPTY OR CURRENT SYMBOL *)
303
304     CURFILENAME,
305     TEMPCURFILENAME : ALFA:         (* CURRENT FILENAME *)
306
307     FILESTATUS,
308     TEMPFILESTATUS  : FILETYPE:     (* NEW OR OLD *)
309
310     FILECHANGES,
311     TEMPFILECHANGES : INTEGER:      (* NUMBER OF CHANGES TO FILE *)
312
313     CURALPHABET,
314     TEMPCURALPHABET : KEYSYMBOL:    (* NORMAL OR ASCII *)
315
316
317     INCORETEXT,
318     TEMPINCORETEXT : INTERNALFILE: (* INTERNAL REPRESENTATION *)
319
320
321     TOPSEQ,
322     UPPERSEQ,
323     HOLE,
324     CURRENTLINE,
325     LOWERSEQ,
326     BOTTOMSEQ,
327
328     TEMPTOPSEQ,
329     TEMPUPPERSEQ,
330     TEMPHOLE,
331     TEMPCURRENTLINE,
332     TEMPLOWERSEQ,
333     TEMPBOTTOMSEQ    : INCOREPTRS: (* POINTERS INTO INCORETEXT *)
334
335 (*$Z *)
```

```
336
337 PROCEDURE PROTECT;
338
339 (*************************************************************************)
340 (*                                                                       *)
341 (* PROTECT IS A UMASS DEPENDANT ROUTINE TO TURN ON THE TERMINAL          *)
342 (* INTERRUPT CAPABILITY. IT CLEARS THE INTERRUPT COUNTER TO ZERO         *)
343 (* AND INHIBITS SYSTEM ERROR EXITS (FOR PFM AND LFM REQUESTS).          *)
344 (*                                                                       *)
345 (*************************************************************************)
346
347 FORTRAN;
348
349
350 FUNCTION INTERRUPT : BOOLEAN;
351
352 (*************************************************************************)
353 (*                                                                       *)
354 (* INTERRUPT IS A UMASS DEPENDANT ROUTINE TO CHECK IF AN INTERRUPT       *)
355 (* WAS SIGNALLED BY THE TERMINAL. IF IT WAS, TRUE IS RETURNED AND THE    *)
356 (* INTERRUPT COUNTER IS RESET TO ZERO. IF NONE WAS SIGNALLED, FALSE      *)
357 (* IS RETURNED.                                                          *)
358 (*                                                                       *)
359 (*************************************************************************)
360
361 FORTRAN;
362
363
364 FUNCTION DISABLE : BOOLEAN;
365
366 (*************************************************************************)
367 (*                                                                       *)
368 (* DISABLE IS A UMASS DEPENDANT ROUTINE TO CHECK IF AN INTERRUPT WAS     *)
369 (* SIGNALLED. THE INTERRUPT COUNTER IS NOT RESET TO ZERO. IF AN          *)
370 (* INTERRUPT WAS SIGNALLED, TRUE IS RETURNED OTHERWISE FALSE.            *)
371 (*                                                                       *)
372 (*************************************************************************)
373
374 FORTRAN;
375
376 (*$Z *)
```

```
377
378 PROCEDURE GETUSER( VAR A : ALFA ):
379
380 (*************************************************************************)
381 (*                                                                       *)
382 (* GETUSER IS A UMASS DEPENDANT ROUTINE TO RETURN THE 7 CHARACTER        *)
383 (* USER NUMBER OF THE INDIVIDUAL CURRENTLY UTILIZING HOPE.               *)
384 (*                                                                       *)
385 (*************************************************************************)
386
387 FORTRAN;
388
389
390 PROCEDURE SETASCII;
391
392 (*************************************************************************)
393 (*                                                                       *)
394 (* SETASCII IS A UMASS DEPENDANT ROUTINE TO SWITCH THE TERMINAL PRINT    *)
395 (* INTO ASCII PRINT MODE TO DISPLAY UPPER- AND LOWER-CASE CHARACTERS.    *)
396 (*                                                                       *)
397 (*************************************************************************)
398
399 FORTRAN;
400
401
402 PROCEDURE SETNORMAL;
403
404 (*************************************************************************)
405 (*                                                                       *)
406 (* SETNORMAL IS A UMASS DEPENDANT ROUTINE TO RESTORE THE TERMINAL        *)
407 (* PRINTER TO NORMAL (64 CHARACTER) PRINT MODE.                          *)
408 (*                                                                       *)
409 (*************************************************************************)
410
411 FORTRAN;
412
413
414 PROCEDURE SETBACK;
415
416 (*************************************************************************)
417 (*                                                                       *)
418 (* SETBACK IS A UMASS DEPENDANT ROUTINE TO RETURN THE TERMINAL           *)
419 (* PRINTER BACK TO THE PRINT MODE IT WAS IN BEFORE HOPE WAS ENTERED.     *)
420 (*                                                                       *)
421 (*************************************************************************)
422
423 FORTRAN;
424
425 (*$Z *)
```

```
426
427 FUNCTION HAVEFILE( FILENAME : ALFA ) : BOOLEAN;
428
429 (*****************************************************************)
430 (*                                                               *)
431 (* HAVEFILE IS A UMASS DEPENDANT ROUTINE TO DETERMINE IF A SPECIFIC   *)
432 (* FILE IS CURRENTLY IN THE HOPE USER'S FILE CATALOG. IF THE FILE IS  *)
433 (* FOUND, TRUE IS RETURNED OTHERWISE FALSE IS RETURNED.          *)
434 (*                                                               *)
435 (*****************************************************************)
436
437 FORTRAN;
438
439
440 PROCEDURE REMOVE( FILENAME : ALFA );
441
442 (*****************************************************************)
443 (*                                                               *)
444 (* REMOVE IS A UMASS DEPENDANT ROUTINE TO REMOVE (IE. 'PURGE') A FILE *)
445 (* FROM THE HOPE USER'S FILE CATALOG.                            *)
446 (*                                                               *)
447 (*****************************************************************)
448
449 FORTRAN;
450
451
452 PROCEDURE NEXTFILE(VAR NAME, DATE : ALFA;
453                    VAR FINISHED : BOOLEAN );
454
455 (*****************************************************************)
456 (*                                                               *)
457 (* NEXTFILE IS A UMASS DEPENDANT CO-ROUTINE WHICH IS REPEATEDLY      *)
458 (* CALLED TO RETURN THE NEXT FILE (NAME AND PACKED DATE OF CREATION) *)
459 (* IN THE CATALOG. THE FIRST CALL TO NEXTFILE WILL INITIATE THE      *)
460 (* PROCESS, REPEATED CALLS WILL RETURN THE FILE INFO ONE AT A TIME.  *)
461 (* WHEN THE LAST FILE IS ENCOUNTERED, FINISHED WILL BE SET TO TRUE.  *)
462 (*                                                               *)
463 (*****************************************************************)
464
465 FORTRAN;
466
467 (*$Z *)
```

```
468
469 PROCEDURE FIND(    FILENAME : ALFA:
470                 VAR SCRATCH : TEXT:
471                 VAR OK : BOOLEAN ):
472
473 (*************************************************************************)
474 (*                                                                       *)
475 (* FIND IS A UMASS DEPENDANT ROUTINE TO RETRIEVE (IE. 'GET') A FILE      *)
476 (* FROM THE HOPE USER'S CATALOG. IF THE FILE IS FOUND (WHICH MUST BE     *)
477 (* AN 'INDIRECT' FILE), IT WILL BE MADE A LOCAL FILE WITH THE NAME       *)
478 (* 'SCRATCH' AND OK WILL BE SET TO TRUE. IF THE FILE IS NOT FOUND        *)
479 (* (OR THE FILE IS 'DIRECT'), OK WILL BE SET TO FALSE.                   *)
480 (*                                                                       *)
481 (* *** NOTE: THIS ROUTINE IS PERFORMING A SURREPTITIOUS FILE SWITCH      *)
482 (*           WITHOUT UTILIZING PASCAL'S SYSTEM ROUTINES.                 *)
483 (*                                                                       *)
484 (*************************************************************************)
485
486 FORTRAN:
487
488
489 PROCEDURE SAVE(     FILENAME : ALFA:
490                 VAR SCRATCH : TEXT:
491                 VAR OK : BOOLEAN ):
492
493 (*************************************************************************)
494 (*                                                                       *)
495 (* SAVE IS A UMASS DEPENDANT ROUTINE TO RETURN (IE. 'REPLACE') THE       *)
496 (* FILE 'SCRATCH' INTO THE HOPE USER'S CATALOG (WITH THE SPECIFIED       *)
497 (* NAME). IF THE FILE CANNOT BE SAVED (TOO MANY FILES, FILE TOO LONG,    *)
498 (* FILE IS 'DIRECT'), OK WILL BE SET TO FALSE OTHERWISE (IF IT IS        *)
499 (* SUCCESSFULL) IT WILL BE SET TO TRUE.                                  *)
500 (*                                                                       *)
501 (*************************************************************************)
502
503 FORTRAN:
504
505 (*$Z *)
```

```
506
507 PROCEDURE INITIALIZE;
508
509 (*************************************************************************)
510 (*                                                                       *)
511 (* INITIALIZE WILL PLACE INITIAL VALUES IN ALL GLOBALS.                  *)
512 (*                                                                       *)
513 (*************************************************************************)
514
515 VAR I : 1..MAXSTRLEN;
516
517     J : 1..MAXSYMLEN;
518
519 BEGIN (* INITIALIZE *)
520
521    KEYWORD [FORWARD   ]  :=  'FORWARD   ';
522    KEYWORD [BACKWARD  ]  :=  'BACKWARD  ';
523    KEYWORD [LIST      ]  :=  'LIST      ';
524    KEYWORD [DELETE    ]  :=  'DELETE    ';
525    KEYWORD [TRANSFER  ]  :=  'TRANSFER  ';
526    KEYWORD [COPY      ]  :=  'COPY      ';
527    KEYWORD [MAKE      ]  :=  'MAKE      ';
528    KEYWORD [INSERT    ]  :=  'INSERT    ';
529    KEYWORD [ASSUME    ]  :=  'ASSUME    ';
530    KEYWORD [SHOW      ]  :=  'SHOW      ';
531    KEYWORD [PRESERVE  ]  :=  'PRESERVE  ';
532    KEYWORD [DESTROY   ]  :=  'DESTROY   ';
533    KEYWORD [UNDO      ]  :=  'UNDO      ';
534    KEYWORD [QUIT      ]  :=  'QUIT      ';
535    KEYWORD [AFTER     ]  :=  'AFTER     ';
536    KEYWORD [BEFORE    ]  :=  'BEFORE    ';
537    KEYWORD [OVER      ]  :=  'OVER      ';
538    KEYWORD [FILESYM   ]  :=  'FILE      ';
539    KEYWORD [MARGIN    ]  :=  'MARGIN    ';
540    KEYWORD [ELLIPSIS  ]  :=  'ELLIPSIS  ';
541    KEYWORD [JOKER     ]  :=  'JOKER     ';
542    KEYWORD [TOP       ]  :=  'TOP       ';
543    KEYWORD [BOTTOM    ]  :=  'BOTTOM    ';
544    KEYWORD [ALPHABET  ]  :=  'ALPHABET  ';
545    KEYWORD [CATALOG   ]  :=  'CATALOG   ';
546    KEYWORD [ALL       ]  :=  'ALL       ';
547    KEYWORD [HERE      ]  :=  'HERE      ';
548    KEYWORD [NULL      ]  :=  'NULL      ';
549    KEYWORD [ASCII     ]  :=  'ASCII     ';
550    KEYWORD [NORMAL    ]  :=  'NORMAL    ';
551    KEYWORD [ISSYM     ]  :=  'IS        ';
552    KEYWORD [INTO      ]  :=  'INTO      ';
553    KEYWORD [FIRST     ]  :=  'FIRST     ';
554    KEYWORD [LAST      ]  :=  'LAST      ';
555    KEYWORD [YES       ]  :=  'YES       ';
556    KEYWORD [NO        ]  :=  'NO        ';
557    KEYWORD [GRIPE     ]  :=  'GRIPE     ';
558    KEYWORD [UNKNOWN   ]  :=  'UNKNOWN   ';
559
560
561    LINELIMIT (TERMNAL, MAXINT);  (* PASCAL 6000-3.4 ROUTINE *)
562    LINELIMIT (SCRATCH, MAXINT);
563    LINELIMIT (COMMENT, MAXINT);
564    LINELIMIT (OUTPUT,  MAXINT);
565
566
567    REWRITE (SCRATCH);
568    REWRITE (COMMENT);
```

```
569
570
571 (* CONSTANTS *)
572
573    FOR I := 1 TO MAXSTRLEN DO
574       NULLTEXT[I] := BLANK:
575
576    FOR J := 1 TO MAXSYMLEN DO
577       NULLSYM.VAL [J] := BLANK:
578
579    NULLSYM.LEN := NULLSYMLEN;
580
581    MAXSEQ.FIRST := 1:
582    MAXSEQ.LAST  := MAXNUMCHARS:
583
584    NULLSEQ.FIRST := NULLPOINTER:
585    NULLSEQ.LAST  := NULLPOINTER:
586
587    WITH NULLPATTERN DO
588       FOR I := 1 TO MAXSTRLEN DO
589          BEGIN
590             SYM [I] := CONSTANT:
591             CHR [I] := BLANK:
592             JOK [I] := BLANK:
593             ELL [I] := NULLSEQ:
594          END:
595
596    LM [1]  :=  OUTMARKER:
597    LM [2]  :=  INMARKER:
598
599    RM [1]  :=  INMARKER:
600    RM [2]  :=  OUTMARKER:
601
602    WITH NULLREQUEST DO
603       BEGIN
604          NAME       := UNKNOWN:
605          REPEATER   := NULLREPEATER:
606          TEXTSTR    := NULLTEXT:
607          NEWTEXTSTR := NULLTEXT:
608          TEXTLEN    := NULLSTRLEN:
609          NEWTEXTLEN := NULLSTRLEN:
610          PLACE      := UNKNOWN:
611          OPTION     := UNKNOWN:
612          VALUE      := UNKNOWN:
613          SPECIALSYM := NULLSYM:
614          FILENAME   := EMPTYFILE:
615       END:
616
617 (*$Z *)
```

```
618
619 (* STATUS VARIABLES *)
620
621     LASTREQUEST := NULLREQUEST;
622
623     CURFILENAME := EMPTYFILE;
624
625     MARGINSYM    := NULLSYM;
626     JOKERSYM     := NULLSYM;
627     ELLIPSISSYM := NULLSYM;
628
629     CURALPHABET := NORMAL;
630
631     SETNORMAL;
632
633     TOPSEQ       := NULLSEQ;
634     UPPERSEQ     := NULLSEQ;
635     HOLE         := MAXSEQ;
636     CURRENTLINE := NULLSEQ;
637     LOWERSEQ     := NULLSEQ;
638     BOTTOMSEQ    := NULLSEQ;
639
640     FILESTATUS := OLD;
641     FILECHANGES := 0;
642
643     ENDOFLINE := TRUE;
644     SHOWINPUT := FALSE;
645     SHOWLINE  := FALSE;
646
647     ERROR := FALSE;
648     DONE  := FALSE;
649
650 END; (* INITIALIZE *)
651
652 (*$Z *)
```

```
653
654 PROCEDURE PRINT( KEY : KEYSYMBOL ):
655
656 (*****************************************************************************)
657 (*                                                                           *)
658 (* PRINT DISPLAYS THE GIVEN KEYWORD (REMOVING ALL BLANKS).                   *)
659 (*                                                                           *)
660 (*****************************************************************************)
661
662 VAR I : 1..MAXKEYLEN:
663
664 BEGIN (* PRINT *)
665
666    I := 1:
667
668    WHILE KEYWORD [KEY] [I]  <>  BLANK DO
669       BEGIN
670          WRITE (KEYWORD [KEY] [I]):
671          I := I + 1:
672       END:
673
674 END: (* PRINT *)
675
676 (*$Z *)
```

```
677
678 PROCEDURE PRINTDATE;
679
680 (*************************************************************************)
681 (*                                                                       *)
682 (* PRINTDATE IS A UMASS DEPENDANT ROUTINE TO CONVERT AND DISPLAY THE     *)
683 (* NUMERIC DATE IN A MORE READABLE FORM THAN IT IS CURRENTLY IN.         *)
684 (*                                                                       *)
685 (*************************************************************************)
686
687 CONST SEPERATOR = ', ';
688       CENTURY = '19';
689
690 VAR MONTH : 1..12;
691     A : ALFA;
692
693 BEGIN (* PRINTDATE *)
694
695    DATE (A);  (* PASCAL 6000-3.4 ROUTINE *)
696
697    MONTH := (ORD(A[5])-ORD('0')) * 10 + (ORD(A[6])-ORD('0'));
698
699    CASE MONTH OF
700
701       1 : WRITE ('JANUARY ');
702       2 : WRITE ('FEBRUARY ');
703       3 : WRITE ('MARCH ');
704       4 : WRITE ('APRIL ');
705       5 : WRITE ('MAY ');
706       6 : WRITE ('JUNE ');
707       7 : WRITE ('JULY ');
708       8 : WRITE ('AUGUST ');
709       9 : WRITE ('SEPTEMBER ');
710      10: WRITE ('OCTOBER ');
711      11: WRITE ('NOVEMBER ');
712      12: WRITE ('DECEMBER ');
713
714    END; (* CASE *)
715
716    WRITE (A[8], A[9], SEPERATOR);
717
718    WRITE (CENTURY, A[2], A[3]);
719
720 END; (* PRINTDATE *)
721
722 (*$Z *)
```

```
836         ELSE
837            WRITE ('AM'):
838
839 END: (* PRINTPACKED *)
840
841 (*$Z *)


842
843 PROCEDURE PRINTERRLINE:
844
845 (*************************************************************************)
846 (*                                                                       *)
847 (* PRINTERRLINE WILL INDICATE THE POINT IN A REQUEST WHICH WAS IN         *)
848 (* ERROR BY AN ERROR POINTER. IF SOMETHING WAS DISPLAYED AFTER THE       *)
849 (* REQUEST WAS ENTERED, THE ENTIRE INPUT LINE IS RE-DISPLAYED BEFORE     *)
850 (* THE ERROR POINTER IS DISPLAYED.                                       *)
851 (*                                                                       *)
852 (* NOTE: THIS ROUTINE HAS BEEN DE-ACTIVATED. (SEE SHOWERROR)             *)
853 (*                                                                       *)
854 (*************************************************************************)
855
856 VAR I : STRINGINDEX:
857
858 BEGIN (* PRINTERRLINE *)
859
860    IF SHOWINPUT
861       THEN
862          BEGIN
863             IF (CURALPHABET = ASCII)
864                THEN
865                   WRITE (PRINTASCII):
866
867             WRITE (MSGPREFIX):
868
869             FOR I := 1 TO LINELENGTH DO
870                WRITE (TEXTLINE [I]);
871
872             WRITELN:
873
874             SHOWINPUT := FALSE:
875          END;
876
877    WRITE (MSGPREFIX):
878
879    IF (CURLINEPOS > 2)
880       THEN
881          BEGIN
882             FOR I := 1 TO (CURLINEPOS - 2) DO
883                IF (TEXTLINE [I] <> BREAKCODE) AND
884                   (TEXTLINE [I] <> ESCAPECODE)
885                   THEN
886                      WRITE (BLANK):
887
888             IF (ENDOFLINE)
889                THEN
890                   WRITE (BLANK):
891          END;
892
893    WRITELN (ERRORPOINTER):
894
895 END: (* PRINTERRLINE *)
896
897 (*$Z *)
```

```
898
899 PROCEDURE SHOWSYNERROR( MSG : ERRORMSG ):
900
901 (*************************************************************************)
902 (*                                                                       *)
903 (* SHOWSYNERROR WILL DISPLAY A MESSAGE CONCERNING A SYNTACTIC ERROR.     *)
904 (*                                                                       *)
905 (*************************************************************************)
906
907 BEGIN (* SHOWSYNERROR *)
908
909    WRITE (MSGPREFIX):
910
911    CASE MSG OF
912
913       ERRISSYM        : BEGIN
914                            WRITE ('KEYWORD ''');
915                            PRINT (ISSYM):
916                            WRITE (''' MISSING IN ');
917                            PRINT (ASSUME):
918                            WRITE (' ');
919                            PRINT (REQUEST.OPTION):
920                            WRITE (' REQUEST ');
921                         END:
922
923       ERRFILENAME     : BEGIN
924                            PRINT (FILESYM):
925                            WRITE (' NAME MISSING IN ');
926                            PRINT (REQUEST.NAME):
927                            WRITE (' ');
928                            IF (REQUEST.NAME IN [ASSUME, INSERT])
929                               THEN
930                                  BEGIN
931                                   PRINT (FILESYM):
932                                   WRITE (' ');
933                                  END:
934                            WRITE ('REQUEST ');
935                         END:
936
937       ERRSPECIALSYM : BEGIN
938                          WRITE ('INVALID SPECIAL SYMBOL OR KEYWORD ''');
939                          PRINT (NULL);
940                          WRITE (''' MISSING IN ');
941                          PRINT (ASSUME):
942                          WRITE (' ');
943                          PRINT (REQUEST.OPTION):
944                          WRITE (' REQUEST ');
945                       END:
946
947       ERRNUMBER       : BEGIN
948                          WRITE ('INVALID REPEATER IN ');
949                          PRINT (REQUEST.NAME):
950                          WRITE (' REQUEST ');
951                       END:
952
953       ERRFIRST        : BEGIN
954                          WRITE ('KEYWORD ''');
955                          PRINT (HERE):
956                          WRITE (''' OR ''');
957                          PRINT (FIRST):
958                          WRITE (''' MISSING IN ');
959                          PRINT (ASSUME):
960                          WRITE (' ');
```

```
961                         PRINT (TOP):
962                         WRITE (' REQUEST '):
963                       END:
964
965       ERRLAST         : BEGIN
966                         WRITE ('KEYWORD '''):
967                         PRINT (HERE):
968                         WRITE (''' OR '''):
969                         PRINT (LAST):
970                         WRITE (''' MISSING IN '):
971                         PRINT (ASSUME):
972                         WRITE (' '):
973                         PRINT (BOTTOM):
974                         WRITE (' REQUEST '):
975                       END:
976
977       ERRNORMAL       : BEGIN
978                         WRITE ('KEYWORD '''):
979                         PRINT (NORMAL):
980                         WRITE (''' OR '''):
981                         PRINT (ASCII):
982                         WRITE (''' MISSING IN '):
983                         PRINT (ASSUME):
984                         WRITE (' '):
985                         PRINT (ALPHABET):
986                         WRITE (' REQUEST '):
987                       END:
988
989       ERRINTO         : BEGIN
990                         WRITE ('KEYWORD '''):
991                         PRINT (INTO):
992                         WRITE (''' MISSING IN '):
993                         PRINT (REQUEST.NAME):
994                         WRITE (' REQUEST '):
995                       END:
996
997       ERRENDREQUEST   : BEGIN
998                         WRITE ('INVALID CHARACTERS IN '):
999                         PRINT (REQUEST.NAME):
1000                        WRITE (' '):
1001                        IF (REQUEST.NAME IN [ASSUME, SHOW])
1002                           THEN
1003                              BEGIN
1004                               PRINT (REQUEST.OPTION):
1005                               WRITE (' '):
1006                              END;
1007                        IF (REQUEST.NAME = INSERT)
1008                           THEN
1009                              BEGIN
1010                               PRINT (REQUEST.PLACE):
1011                               WRITE (' '):
1012                              END:
1013                        WRITE ('REQUEST '):
1014                      END:
1015
1016      ERRNOREQUEST    : IF TEXTLINE[CURLINEPOS] = KEYWORD[PRESERVE][1]
1017                           THEN
1018                              BEGIN
1019                                 PRINT (PRESERVE):
1020                                 WRITE (' REQUEST CANNOT BE ABBREVIATED'):
1021                              END
1022                           ELSE
1023                              IF TEXTLINE[CURLINEPOS]=KEYWORD[DESTROY][1]
1024                                 THEN
```

```
1025                                    BEGIN
1026                                     PRINT (DESTROY):
1027                                     WRITE (' REQUEST CANNOT'):
1028                                     WRITE (' BE ABBREVIATED'):
1029                                    END
1030                                   ELSE
1031                                    BEGIN
1032                                     WRITE ('INVALID FIRST CHARACTER '):
1033                                     WRITE ('FOR REQUEST NAME '):
1034                                    END:
1035
1036        ERROPTION      : BEGIN
1037                            WRITE ('INVALID OPTION FOR '):
1038                            PRINT (ASSUME):
1039                            WRITE (' REQUEST '):
1040                         END:
1041
1042        ERRMARGINSYM   : BEGIN
1043                            WRITE ('LEFT OR RIGHT '):
1044                            PRINT (MARGIN):
1045                            WRITE (' SYMBOL MISSING IN NEW TEXT STRING'):
1046                         END;
1047
1048        ERRELLIPSISSYM : BEGIN
1049                            WRITE ('EXCESS '):
1050                            PRINT (ELLIPSIS):
1051                            WRITE (' SYMBOLS IN NEW TEXT STRING '):
1052                         END:
1053
1054        ERRJOKERSYM    : BEGIN
1055                            WRITE ('EXCESS '):
1056                            PRINT (JOKER):
1057                            WRITE (' SYMBOLS IN NEW TEXT STRING ');
1058                         END:
1059
1060        ERRELLPOSITION : BEGIN
1061                            PRINT (ELLIPSIS):
1062                            WRITE (' SYMBOL CANNOT BE FIRST'):
1063                            WRITE (' OR LAST ELEMENT IN TEXT STRING'):
1064                         END;
1065
1066        ERRASCIIJOKER  : BEGIN
1067                            WRITE ('CANNOT USE '):
1068                            PRINT (JOKER):
1069                            WRITE (' SYMBOL WITH '):
1070                            PRINT (ASCII):
1071                            WRITE (' '):
1072                            PRINT (ALPHABET):
1073                         END:
1074
1075    END: (* CASE *)
1076
1077    WRITELN:
1078
1079 END: (* SHOWSYNERROR *)
1080
1081 (*$7 *)
```

```
1082
1083 PROCEDURE SHOWSEMERROR( MSG : ERRORMSG );
1084
1085 (*************************************************************************)
1086 (*                                                                       *)
1087 (* SHOWSEMERROR WILL DISPLAY A MESSAGE CONCERNING A SEMANTIC ERROR.      *)
1088 (*                                                                       *)
1089 (*************************************************************************)
1090
1091 BEGIN (* SHOWSEMERROR *)
1092
1093    WRITE (MSGPREFIX);
1094
1095    CASE MSG OF
1096
1097       ERRCANTSHOW     : BEGIN
1098                            WRITE ('CANNOT ');
1099                            PRINT (SHOW);
1100                            WRITE (' ');
1101                            PRINT (REQUEST.OPTION);
1102                            WRITE (' WITHOUT A CURRENT ');
1103                            PRINT (FILESYM);
1104                         END;
1105
1106       ERRUNDO         : IF (LASTREQUEST.NAME = UNKNOWN) OR
1107                            (LASTREQUEST.NAME = UNDO) OR
1108                            (CURFILENAME = EMPTYFILE)
1109                            THEN
1110                               BEGIN
1111                                  WRITE ('THERE IS NOTHING TO ');
1112                                  PRINT (UNDO);
1113                               END
1114                            ELSE
1115                               BEGIN
1116                                  WRITE ('CANNOT ');
1117                                  PRINT (UNDO);
1118                                  WRITE (' ');
1119                                  PRINT (LASTREQUEST.NAME);
1120                                  WRITE (' ');
1121                                  WRITE ('REQUEST ');
1122                               END;
1123
1124       ERRNOTFOUND     : BEGIN
1125                            IF (MATCHNUM = REQUEST.REPEATER)
1126                               THEN
1127                                  WRITE ('NO')
1128                               ELSE
1129                                  BEGIN
1130                                   WRITE ('ONLY ');
1131                                   WRITE (REQUEST.REPEATER-MATCHNUM : 1);
1132                                  END;
1133                            IF (REQUEST.TEXTLEN = NULLSTRLEN)
1134                               THEN
1135                                  WRITE (' LINE(S) FOUND FOR ')
1136                               ELSE
1137                                  BEGIN
1138                                   WRITE (' OCCURENCE(S) OF TEXT ');
1139                                   WRITE ('STRING FOUND FOR ');
1140                                  END;
1141                            PRINT (REQUEST.NAME);
1142                            WRITE (' REQUEST ');
1143                         END;
1144
```

```
1145        ERRNOCURFILE   : BEGIN
1146                            PRINT (REQUEST.NAME):
1147                            WRITE (' '):
1148                            IF (REQUEST.NAME IN [ASSUME, SHOW])
1149                               THEN
1150                                  BEGIN
1151                                   PRINT (REQUEST.OPTION):
1152                                   WRITE (' '):
1153                                  END:
1154                               WRITE ('REQUEST CANNOT BE PERFORMED '):
1155                               WRITE ('WITHOUT A CURRENT '):
1156                               PRINT (FILESYM):
1157                             END:
1158
1159        ERRNOTALLLINES : BEGIN
1160                            IF (LINESFOUND = 0)
1161                               THEN
1162                                  WRITE ('NO ')
1163                               ELSE
1164                                  BEGIN
1165                                   WRITE ('ONLY '):
1166                                   WRITE (LINESFOUND : 1):
1167                                   WRITE (' '):
1168                                  END:
1169                               WRITE ('LINE(S) WILL FIT INTO '):
1170                               WRITE ('CURRENT '):
1171                               PRINT (FILESYM):
1172                             END:
1173
1174        ERRCANTSAVE    : BEGIN
1175                            WRITE ('CANNOT '):
1176                            PRINT (REQUEST.NAME):
1177                            WRITE (' '):
1178                            IF (REQUEST.NAME IN [TRANSFER, COPY])
1179                               THEN
1180                                  BEGIN
1181                                   PRINT (INTO):
1182                                   WRITE (' '):
1183                                  END:
1184                               PRINT (FILESYM):
1185                               WRITE (' '):
1186                               WRITE (REQUEST.FILENAME):
1187                             END:
1188
1189        ERRNOROOM      : BEGIN
1190                            WRITE ('CURRENT '):
1191                            PRINT (FILESYM):
1192                            WRITE (' TOO FULL TO COMPLETE '):
1193                            PRINT (REQUEST.NAME):
1194                            WRITE (' REQUEST'):
1195                          END:
1196
1197        ERRMISSINGTEXT : BEGIN
1198                            WRITE ('TEXT STRING MISSING IN '):
1199                            PRINT (MAKE):
1200                            WRITE (' REQUEST'):
1201                          END:
1202
1203        ERRTEXTLENGTH  : BEGIN
1204                            WRITE ('TEXT STRING IS '):
1205                            WRITE ('TOO LONG TO RECOGNIZE '):
1206                          END:
1207
1208        ERRCHARACTERS  : BEGIN
```

```
1209                            WRITE ('THE CHARACTERS ''');
1210                            IF (CURALPHABET <> ASCII)
1211                               THEN
1212                                  WRITE (INMARKER)
1213                               ELSE
1214                                  WRITE (ASCINMARKER);
1215                            WRITE (''' AND ''');
1216                            IF (CURALPHABET <> ASCII)
1217                               THEN
1218                                  WRITE (OUTMARKER)
1219                               ELSE
1220                                  WRITE (ASCOUTMARKER);
1221                            WRITE (''' CANNOT OCCUR IN ');
1222                            WRITE ('SEQUENCE IN A TEXT STRING ');
1223                         END;
1224
1225       ERRSTOPREQUEST : BEGIN
1226                            WRITE ('INTERRUPT ');
1227                            PRINT (REQUEST.NAME);
1228                            WRITE (' REQUEST');
1229                         END;
1230
1231    END; (* CASE *)
1232
1233    WRITELN;
1234
1235 END; (* SHOWSEMERROR *)
1236
1237 (*$Z *)
```

```
1238
1239 PROCEDURE SHOWVALUE( SYM : VALUESYM ):
1240
1241 (*****************************************************************)
1242 (*                                                               *)
1243 (* SHOWVALUE WILL DISPLAY A MESSAGE ABOUT A PARTICULAR VALUE OF  *)
1244 (* THE CURRENT STATE OF HOPE.                                    *)
1245 (*                                                               *)
1246 (*****************************************************************)
1247
1248 BEGIN (* SHOWVALUE *)
1249
1250    WRITE (MSGPREFIX):
1251
1252    CASE SYM OF
1253
1254       VALFILENAME   : IF (CURFILENAME <> EMPTYFILE)
1255                       THEN
1256                          BEGIN
1257                             WRITE ('CURRENT '):
1258                             PRINT (FILESYM):
1259                             WRITE (' NAME IS '):
1260                             WRITE (CURFILENAME):
1261                          END
1262                       ELSE
1263                          BEGIN
1264                             WRITE ('THERE IS '):
1265                             WRITE ('NO CURRENT '):
1266                             PRINT (FILESYM):
1267                          END;
1268
1269       VALMARGINSYM : BEGIN
1270                         PRINT (MARGIN):
1271                         WRITE (' SYMBOL IS '):
1272                         IF (MARGINSYM.VAL <> NULLSYM.VAL)
1273                            THEN
1274                               WRITE (MARGINSYM.VAL)
1275                            ELSE
1276                               PRINT (NULL):
1277                      END:
1278
1279       VALJOKERSYM  : BEGIN
1280                         PRINT (JOKER):
1281                         WRITE (' SYMBOL IS '):
1282                         IF (JOKERSYM.VAL <> NULLSYM.VAL)
1283                            THEN
1284                               WRITE (JOKERSYM.VAL)
1285                            ELSE
1286                               PRINT (NULL):
1287                      END:
1288
1289       VALELLIPSIS  : BEGIN
1290                         PRINT (ELLIPSIS):
1291                         WRITE (' SYMBOL IS '):
1292                         IF (ELLIPSISSYM.VAL <> NULLSYM.VAL)
1293                            THEN
1294                               WRITE (ELLIPSISSYM.VAL)
1295                            ELSE
1296                               PRINT (NULL):
1297                      END:
1298
1299       VALALPHABET  : BEGIN
1300                         PRINT (ALPHABET):
```

```
1301                            WRITE (' IS '):
1302                            PRINT (CURALPHABET):
1303                         END:
1304
1305       VALTOP         : BEGIN
1306                            PRINT (TOP):
1307                            WRITE (' OF '):
1308                            PRINT (FILESYM):
1309                            WRITE (' IS '):
1310                            IF (CURRENTLINE.FIRST <> NULLPOINTER)
1311                               THEN
1312                                  IF (TOPSEQ.LAST = NULLPOINTER)
1313                                   THEN
1314                                    BEGIN
1315                                     PRINT (FIRST):
1316                                     WRITE (' LINE OF '):
1317                                     PRINT (FILESYM):
1318                                    END:
1319                         END:
1320
1321       VALCURLINE     : BEGIN
1322                            WRITE ('CURRENT LINE IS '):
1323                         END:
1324
1325       VALBOTTOM      : BEGIN
1326                            PRINT (BOTTOM):
1327                            WRITE (' OF '):
1328                            PRINT (FILESYM):
1329                            WRITE (' IS '):
1330                            IF (CURRENTLINE.FIRST <> NULLPOINTER)
1331                               THEN
1332                                  IF (BOTTOMSEQ.FIRST = NULLPOINTER)
1333                                   THEN
1334                                    BEGIN
1335                                     PRINT (LAST):
1336                                     WRITE (' LINE OF '):
1337                                     PRINT (FILESYM):
1338                                    END:
1339                          END:
1340
1341     END: (* CASE *)
1342
1343     WRITELN:
1344
1345     SHOWINPUT := TRUE:
1346
1347 END: (* SHOWVALUE *)
1348
1349 (*$Z *)
```

```
1350
1351 PROCEDURE SHOWMESSAGE( MSG : INFOMSG ):
1352
1353 (*************************************************************************)
1354 (*                                                                       *)
1355 (* SHOWMESSAGE WILL DISPLAY A MESSAGE CONCERNING A PARTICULAR            *)
1356 (* ACTION BEING TAKEN BY HOPE.                                           *)
1357 (*                                                                       *)
1358 (*************************************************************************)
1359
1360 VAR A : ALFA:
1361     I : INTEGER:
1362
1363 BEGIN (* SHOWMESSAGE *)
1364
1365    IF (MSG = INTRO)
1366       THEN
1367          WRITELN:
1368
1369    WRITE (MSGPREFIX):
1370
1371    CASE MSG OF
1372
1373       INTRO       : BEGIN
1374                        WRITE ('WELCOME TO HOPE  '):
1375                        PRINTDATE:
1376                        WRITE ('  '):
1377                        PRINTTIME:
1378                     END:
1379
1380       FINAL       : BEGIN
1381                        WRITE ('END OF HOPE SESSION  '):
1382                        PRINTTIME:
1383                     END:
1384
1385       UNDONE      : BEGIN
1386                        PRINT (LASTREQUEST.NAME):
1387                        IF (LASTREQUEST.NAME = ASSUME)
1388                           THEN
1389                              BEGIN
1390                                 WRITE (' '):
1391                                 PRINT (LASTREQUEST.OPTION):
1392                              END:
1393                        WRITE (' REQUEST UNDONE '):
1394                     END:
1395
1396       NUMLINES    : BEGIN
1397                        WRITE ('MATCHED '):
1398                        WRITE (LINESFOUND : 1):
1399                        WRITE (' LINE(S) '):
1400                        IF (REQUEST.TEXTLEN <> NULLSTRLEN)
1401                           THEN
1402                              WRITE ('WITH TEXT STRING '):
1403                        WRITE ('FOR '):
1404                        PRINT (REQUEST.NAME):
1405                        WRITE (' '):
1406                        IF (REQUEST.NAME IN [TRANSFER, COPY])
1407                           THEN
1408                              BEGIN
1409                                 PRINT (INTO):
1410                                 WRITE (' '):
1411                                 WRITE (REQUEST.FILENAME):
1412                              END:
```

```
1413                    END;
1414
1415        CANTUNDO    : BEGIN
1416                        WRITE ('CANNOT ');
1417                        PRINT (UNDO);
1418                        WRITE (' ');
1419                        PRINT (FILESYM);
1420                        WRITE (' ');
1421                        WRITE (LASTREQUEST.FILENAME);
1422                      END;
1423
1424        NOFILEFOUND : BEGIN
1425                        PRINT (FILESYM);
1426                        WRITE (' ');
1427                        I := 1;
1428                        WHILE (REQUEST.FILENAME [I] <> BLANK) DO
1429                           BEGIN
1430                              WRITE (REQUEST.FILENAME[I]);
1431                              I := I + 1;
1432                           END;
1433                        WRITE (' NOT FOUND FOR ');
1434                        PRINT (REQUEST.NAME);
1435                        WRITE (' REQUEST ');
1436                      END;
1437
1438        ONLYNOYES   : BEGIN
1439                        WRITE ('PLEASE ANSWER WITH ');
1440                        PRINT (YES);
1441                        WRITE (' OR ');
1442                        PRINT (NO);
1443                      END;
1444
1445        NOTDONE     : WRITE ('REQUEST NOT PERFORMED ');
1446
1447        REENTER     : WRITE ('RE-ENTER THE LAST LINE ');
1448
1449        LONGLINE    : BEGIN
1450                        WRITE ('INPUT LINE LONGER THAN ');
1451                        WRITE (MAXSTRLEN : 1);
1452                        WRITE (' CHARACTERS');
1453                      END;
1454
1455        NEWFILE     : BEGIN
1456                        WRITE ('A NEW ');
1457                        PRINT (FILESYM);
1458                        WRITE (' ');
1459                        I := 1;
1460                        WHILE (REQUEST.FILENAME [I] <> BLANK) DO
1461                           BEGIN
1462                              WRITE (REQUEST.FILENAME[I]);
1463                              I := I + 1;
1464                           END;
1465                        WRITE (' HAS BEEN CREATED FOR YOU ');
1466                      END;
1467
1468        RESTRICTION : WRITE ('IMPLEMENTATION RESTRICTION ');
1469
1470        ENTER       : WRITE ('ENTER NEW TEXT ');
1471
1472        FILEINSERT  : BEGIN
1473                        WRITE ('INSERT ');
1474                        PRINT (FILESYM);
1475                        WRITE (' ');
1476                        I := 1;
```

```
1477                            WHILE (REQUEST.FILENAME [I] <> BLANK) DO
1478                               BEGIN
1479                                  WRITE (REQUEST.FILENAME [I]):
1480                                  I := I + 1:
1481                               END:
1482                            WRITE (' '):
1483                            PRINT (REQUEST.PLACE):
1484                            WRITE (' CURRENT LINE'):
1485                         END:
1486
1487       ENDTEXT      : WRITE ('END OF NEW TEXT '):
1488
1489       CATLIST      : BEGIN
1490                            PRINT (CATALOG):
1491                            WRITE (' OF '):
1492                            GETUSER (A):
1493                            WRITE (A):
1494                         END:
1495
1496       FILEHEAD     : BEGIN
1497                            PRINT (FILESYM):
1498                            WRITE (' NAME    LAST '):
1499                            PRINT (PRESERVE):
1500                            WRITE (' DATE'):
1501                         END:
1502
1503       EMPTYCAT     : BEGIN
1504                            WRITE ('EMPTY '):
1505                            PRINT (CATALOG):
1506                         END:
1507
1508       ENTERGRIPE : BEGIN
1509                            WRITE ('PLEASE ENTER YOUR '):
1510                            PRINT (GRIPE):
1511                         END:
1512
1513       GRIPESAVED : BEGIN
1514                            WRITE ('THANK-YOU, YOUR '):
1515                            PRINT (GRIPE):
1516                            WRITE (' HAS BEEN NOTED'):
1517                         END:
1518
1519    END: (* CASE *)
1520
1521    WRITELN:
1522
1523    IF (MSG = FINAL) OR
1524       (MSG = ENTER) OR
1525       (MSG = ENTERGRIPE)
1526       THEN
1527          WRITELN:
1528
1529    IF (MSG <> INTRO)
1530       THEN
1531          SHOWINPUT := TRUE:
1532
1533 END: (* SHOWMESSAGE *)
1534
1535 (*97 *)
```

```
1536
1537 PROCEDURE SHOWERROR( MSG : ERRORMSG ):
1538
1539 (*****************************************************************************)
1540 (*                                                                           *)
1541 (* SHOWERROR WILL DISPLAY A MESSAGE CONCERNING AN ERROR IN THE               *)
1542 (* USER'S REQUEST. 'ERROR' IS SET TO TRUE.                                   *)
1543 (*                                                                           *)
1544 (*****************************************************************************)
1545
1546 VAR I   : STRINGINDEX:
1547     CH : CHAR:
1548
1549 BEGIN (* SHOWERROR *)
1550
1551    ERROR := TRUE:
1552
1553 (* PRINTERRLINE: *)
1554
1555    IF (MSG IN [ERRISSYM..ERRASCIIJOKER])
1556       THEN
1557          SHOWSYNERROR( MSG )
1558       ELSE
1559          SHOWSEMERROR( MSG ):
1560
1561    SHOWINPUT := TRUE:
1562
1563 END: (* SHOWERROR *)
1564
1565 (*$Z *)
```

```
1566
1567 PROCEDURE SHOWLISTING;
1568
1569 (*************************************************************************)
1570 (*                                                                       *)
1571 (* SHOWLISTING WILL DISPLAY THE LINES OF TEXT STORED ON THE SCRATCH      *)
1572 (* FILE ONTO THE TERMINAL IN A READABLE FORM. IF THE FILE IS EMPTY,      *)
1573 (* A MESSAGE WILL BE DISPLAYED.                                          *)
1574 (*                                                                       *)
1575 (*************************************************************************)
1576
1577 VAR CHR : CHAR;
1578
1579 BEGIN (* SHOWLISTING *)
1580
1581    RESET (SCRATCH);
1582
1583    IF (SHOWLINE)
1584       THEN
1585          SHOWVALUE( VALCURLINE );
1586
1587    IF NOT VALIDINFO
1588       THEN
1589          WRITELN (MSGPREFIX, 'EMPTY')
1590       ELSE
1591          WHILE NOT EOF(SCRATCH) AND NOT(DISABLE) DO
1592             BEGIN
1593
1594                WRITE (LSTPREFIX);
1595
1596                WHILE NOT EOLN(SCRATCH) DO
1597                   BEGIN
1598                      READ (SCRATCH, CHR);
1599                      WRITE (CHR);
1600                   END;
1601
1602                READLN (SCRATCH);
1603                WRITELN;
1604
1605             END; (* WHILE *)
1606
1607    SHOWINPUT := TRUE;
1608    VALIDINFO := FALSE;
1609
1610 END; (* SHOWLISTING *)
1611
1612 (*$7 *)
```

```
1613
1614 PROCEDURE GETLINE(    PROMPT : DOUBLECHAR:
1615                   VAR LINE   : STRING:
1616                   VAR LENGTH : STRINGINDEX):
1617
1618 (*****************************************************************)
1619 (*                                                               *)
1620 (* GETLINE IS A UMASS DEPENDANT ROUTINE TO READ AN INPUT LINE FROM    *)
1621 (* THE TERMINAL. THE SPECIFIED PROMPT CHARACTERS ARE DISPLAYED AND A  *)
1622 (* LINE IS REQUESTED. ALL EDITING CHARACTERS (BACKSPACE AND DELETE)   *)
1623 (* ARE REMOVED FROM THE LINE. THE ROUTINE WILL CONTINUE TO REQUEST    *)
1624 (* INPUT UNTIL A NON-BLANK LINE IS ENTERED.                           *)
1625 (*                                                                    *)
1626 (* *** NOTE : IF ALPHABET IS NORMAL, ALL CHARACTERS CONVERTED         *)
1627 (*            FROM ASCII TO STANDARD CHARACTER SET.                   *)
1628 (*                                                                    *)
1629 (*****************************************************************)
1630
1631 VAR FOUND,
1632     DELETELINE : BOOLEAN:
1633
1634     CHR : CHAR:
1635     CHARTYPE : 0..6:
1636
1637 BEGIN (* GETLINE *)
1638
1639    REPEAT
1640
1641       PROTECT:
1642
1643       WRITELN (PROMPT, NOLINEFEED):
1644       WRITELN (TRANSPARENT):
1645       RESET (TERMINAL):
1646
1647       LINE := NULLTEXT:
1648       LENGTH := NULLSTRLEN:
1649       DELETELINE := FALSE:
1650
1651       WHILE NOT(EOF(TERMINAL) OR EOLN(TERMINAL) OR DELETELINE OR DISABLE)
1652          DO
1653          IF (LENGTH >= MAXSTRLEN)
1654             THEN
1655                BEGIN
1656                   SHOWMESSAGE( RESTRICTION ):
1657                   SHOWMESSAGE( LONGLINE ):
1658                   DELETELINE := TRUE:
1659                END
1660             ELSE
1661                BEGIN
1662                   READ (TERMINAL, CHR):
1663
1664                   IF (CHR <> BREAKCODE) AND
1665                      (CHR <> ESCAPECODE)
1666                      THEN
1667                         BEGIN
1668                            LENGTH := LENGTH + 1:
1669                            LINE [LENGTH] := CHR:
1670                         END
1671                      ELSE
1672                         BEGIN
1673
1674                            CHARTYPE := 0:
1675
```

```
1676                    IF (CURALPHABET = ASCII) AND
1677                       (CHR = BREAKCODE)
1678                       THEN
1679                        IF (TERMNAL^ <= LSTPRINT)
1680                         THEN
1681                          CHARTYPE := 1:
1682
1683                    IF (CURALPHABET = ASCII) AND
1684                       (CHR = ESCAPECODE)
1685                       THEN
1686                        IF (TERMNAL^ = CLNCHAR) OR
1687                           (TERMNAL^ = ESCCHAR) OR
1688                           (TERMNAL^ = BRKCHAR)
1689                         THEN
1690                          CHARTYPE := 1:
1691
1692                    IF (CURALPHABET = NORMAL) AND
1693                       (CHR = ESCAPECODE)
1694                       THEN
1695                        IF (TERMNAL^ = CLNCHAR)
1696                         THEN
1697                          BEGIN
1698                           LENGTH := LENGTH + 1:
1699                           LINE [LENGTH] := COLON:
1700                           CHARTYPE := 6:
1701                          END:
1702
1703                    IF (CURALPHABET = NORMAL) AND
1704                       (CHR = ESCAPECODE)
1705                       THEN
1706                        IF (TERMNAL^ = ESCCHAR) OR
1707                           (TERMNAL^ = BRKCHAR)
1708                         THEN
1709                          CHARTYPE := 2:
1710
1711                    IF (CURALPHABET = NORMAL) AND
1712                       (CHR = BREAKCODE) AND
1713                       (TERMNAL^ IN ['A'..'Z'])
1714                       THEN
1715                        CHARTYPE := 3:
1716
1717                    IF (TERMNAL^ = DELCHAR)
1718                       THEN
1719                        CHARTYPE := 4:
1720
1721                    IF (TERMNAL^ = BSCHAR)
1722                       THEN
1723                        CHARTYPE := 5:
1724
1725                    IF (CHARTYPE = 0)
1726                       THEN
1727                        CHARTYPE := 6:
1728
1729                    IF (CHR = ESCAPECODE) AND
1730                       (TERMNAL^ = ESCAPECODE)
1731                       THEN
1732                        BEGIN
1733                         DELETELINE := TRUE:
1734                         WRITELN:
1735                         SHOWERROR( ERRCHARACTERS ):
1736                        END:
1737
1738                    IF (CHR = BREAKCODE) AND
1739                       (TERMNAL^ = BREAKCODE)
```

```
1740                    THEN
1741                     BEGIN
1742                      DELETELINE := TRUE;
1743                      WRITELN;
1744                      SHOWERROR( ERRCHARACTERS );
1745                     END;
1746
1747                  CASE CHARTYPE OF
1748
1749                     1 : IF (LENGTH > MAXSTRLEN - 1)
1750                          THEN
1751                           LENGTH := MAXSTRLEN
1752                          ELSE
1753                           BEGIN
1754                            LENGTH := LENGTH + 1;
1755                            LINE [LENGTH] := CHR;
1756                            READ (TERMINAL, CHR);
1757                            LENGTH := LENGTH + 1;
1758                            LINE [LENGTH] := CHR;
1759                           END;
1760
1761                     2 : BEGIN
1762                          READ (TERMINAL, CHR);
1763                          IF (CHR = ESCCHAR)
1764                           THEN
1765                            CHR := ESCAPECODE
1766                           ELSE
1767                            CHR := BREAKCODE;
1768                          LENGTH := LENGTH + 1;
1769                          LINE [LENGTH] := CHR;
1770                         END;
1771
1772                     3 : BEGIN
1773                          READ (TERMINAL, CHR);
1774                          LENGTH := LENGTH + 1;
1775                          LINE [LENGTH] := CHR;
1776                         END;
1777
1778                     4 : BEGIN
1779                          DELETELINE := TRUE;
1780                          WRITELN;
1781                         END;
1782
1783                     5 : BEGIN
1784                          READ (TERMINAL, CHR);
1785                          IF (LENGTH <> NULLSTRLEN)
1786                           THEN
1787                            IF (LENGTH = 1)
1788                             THEN
1789                              LENGTH := NULLSTRLEN
1790                             ELSE
1791                              IF (CURALPHABET = ASCII) AND
1792                                 ((LINE[LENGTH-1] = BREAKCODE)
1793                                 OR
1794                                 (LINE[LENGTH-1] = ESCAPECODE))
1795                               THEN
1796                                LENGTH := LENGTH - 2
1797                               ELSE
1798                                LENGTH := LENGTH - 1;
1799                         END;
1800
1801                     6 : READ (TERMINAL, CHR);
1802
1803                  END; (* CASE *)
```

```
1804
1805                              END:
1806
1807                    END: (* WHILE IF *)
1808
1809
1810          IF (LENGTH <> NULLSTRLEN)
1811             THEN
1812                BEGIN
1813
1814                   IF (LENGTH MOD 10 = 0) AND
1815                      (LINE [LENGTH] = LINEMARK)
1816                      THEN
1817                         LENGTH := LENGTH - 1:
1818
1819                   IF (DELETELINE)
1820                      THEN
1821                         LENGTH := NULLSTRLEN:
1822
1823                   FOUND := FALSE:
1824
1825                   WHILE (LENGTH <> NULLSTRLEN) AND NOT(FOUND) DO
1826                      IF (LINE [LENGTH] <> LINEMARK)
1827                         THEN
1828                            FOUND := TRUE
1829                         ELSE
1830                            LENGTH := LENGTH - 1:
1831
1832                   FOUND := FALSE:
1833
1834                   WHILE (LENGTH <> NULLSTRLEN) AND NOT(FOUND) DO
1835                      IF (LINE [LENGTH] <> BLANK)
1836                         THEN
1837                            FOUND := TRUE
1838                         ELSE
1839                            LENGTH := LENGTH - 1:
1840
1841                END:
1842
1843       UNTIL NOT(DELETELINE):
1844
1845    END: (* GETLINE *)
1846
1847    (*$7 *)
```

```
1848
1849 PROCEDURE GETCHAR;
1850
1851 (*****************************************************************************)
1852 (*                                                                           *)
1853 (* GETCHAR PLACES THE NEXT CHARACTER FROM THE INPUT STRING 'TEXTLINE' *)
1854 (* INTO 'NEXTCHAR'. IF 'EOLN' IS ENCOUNTERED, 'ENDOFLINE' IS SET.     *)
1855 (*                                                                           *)
1856 (*****************************************************************************)
1857
1858 BEGIN (* GETCHAR *)
1859
1860    IF (CURLINEPOS < LINELENGTH)
1861       THEN
1862          BEGIN
1863             CURLINEPOS := CURLINEPOS + 1;
1864             NEXTCHAR := TEXTLINE [CURLINEPOS];
1865          END
1866       ELSE
1867          BEGIN
1868             ENDOFLINE := TRUE;
1869             NEXTCHAR := BLANK;
1870          END;
1871
1872 END; (* GETCHAR *)
1873
1874 (*$7 *)
```

```
1875
1876 FUNCTION MATCH( KEY : KEYSYMBOL ) : BOOLEAN;
1877
1878 (*****************************************************************)
1879 (*                                                               *)
1880 (*  MATCH ATTEMPTS TO MATCH THE CHARACTER STRING REPRESENTATION OF *)
1881 (*  THE GIVEN KEYWORD. IF THE INPUT STRING MATCHES THE COMPLETE   *)
1882 (*  SPELLING IN THE KEYWORD TABLE, THE FUNCTION RETURNS TRUE AND  *)
1883 (*  UPDATES THE CURRENT LINE POSITION (AND THE NEXT CHARACTER.)   *)
1884 (*  OTHERWISE, IF THE FIRST CHARACTER MATCHES THE NEXT CHARACTER  *)
1885 (*  IN THE INPUT STRING, TRUE IS RETURNED AND THE NEXT CHARACTER  *)
1886 (*  IS UPDATED.                                                  *)
1887 (*                                                               *)
1888 (*  *** NOTE :  UPPER/LOWER CASE ARE ALL CONVERTED TO UPPER CASE. *)
1889 (*                                                               *)
1890 (*****************************************************************)
1891
1892 VAR INDEX   : 1..MAXKEYLEN;
1893
1894     FIRST,
1895     LENGTH  : STRINGINDEX;
1896
1897     HITLONG,
1898     HITSHORT : BOOLEAN;
1899
1900 BEGIN (* MATCH *)
1901
1902    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
1903       GETCHAR;
1904
1905    LENGTH := CURLINEPOS;
1906    FIRST := CURLINEPOS;
1907
1908    INDEX := 1;
1909    HITSHORT := FALSE;
1910    HITLONG := TRUE;
1911
1912
1913    WHILE (KEYWORD [KEY] [INDEX] <> BLANK) AND HITLONG DO
1914       IF (LENGTH > LINELENGTH)
1915          THEN
1916             HITLONG := FALSE
1917          ELSE
1918             BEGIN
1919                IF (LENGTH < LINELENGTH)
1920                   THEN
1921                      IF (TEXTLINE [LENGTH] = BREAKCODE) AND
1922                         (TEXTLINE [LENGTH+1] IN ['A'..'Z']) AND
1923                         (CURALPHABET = ASCII)
1924                         THEN
1925                            LENGTH := LENGTH + 1;
1926
1927                HITLONG := (TEXTLINE [LENGTH] = KEYWORD [KEY] [INDEX]);
1928                INDEX := INDEX + 1;
1929                LENGTH := LENGTH + 1;
1930             END;
1931
1932
1933    IF (KEY <> PRESERVE) AND
1934       (KEY <> DESTROY)
1935       THEN
1936          BEGIN
1937             IF (FIRST < LINELENGTH)
```

```
1938                  THEN
1939                     IF (TEXTLINE [FIRST] = BREAKCODE) AND
1940                        (TEXTLINE [FIRST+1] IN ['A'..'Z']) AND
1941                        (CURALPHABET = ASCII)
1942                        THEN
1943                           FIRST := FIRST + 1;
1944
1945               HITSHORT  :=  (TEXTLINE [FIRST] = KEYWORD [KEY] [1]);
1946            END;
1947
1948
1949     IF HITLONG
1950        THEN
1951           BEGIN
1952              CURLINEPOS := LENGTH - 1;
1953              GETCHAR;
1954           END
1955        ELSE
1956           IF HITSHORT
1957              THEN
1958                 BEGIN
1959                    CURLINEPOS := FIRST;
1960                    GETCHAR;
1961                 END;
1962
1963     MATCH := (HITSHORT OR HITLONG);
1964
1965  END; (* MATCH *)
1966
1967  (*$7 *)
```

```
1968
1969 PROCEDURE GETRESPONSE( VAR ANSWER : KEYSYMBOL ):
1970
1971 (*************************************************************************)
1972 (*                                                                       *)
1973 (* GETRESPONSE WILL READ AN INPUT LINE AND ATTEMPT TO MATCH EITHER       *)
1974 (* 'YES' OR 'NO'. IF FOUND, THE ANSWER IS SET APPROPRIATELY. IF NOT      *)
1975 (* FOUND, THE ROUTINE WILL CONTINUE TO REQUEST INPUT UNTIL 'YES' OR      *)
1976 (* 'NO' IS ENTERED.                                                      *)
1977 (*                                                                       *)
1978 (*************************************************************************)
1979
1980 VAR
1981     TEMPNEXTCHAR    : CHAR:
1982
1983     TEMPTEXTLINE    : STRING:
1984
1985     TEMPCURLINEPOS,
1986     TEMPLINELENGTH  : STRINGINDEX:
1987
1988     TEMPENDOFLINE   : BOOLEAN:
1989
1990 BEGIN (* GETRESPONSE *)
1991
1992    TEMPNEXTCHAR    := NEXTCHAR:
1993    TEMPTEXTLINE    := TEXTLINE:
1994    TEMPCURLINEPOS  := CURLINEPOS:
1995    TEMPLINELENGTH  := LINELENGTH:
1996    TEMPENDOFLINE   := ENDOFLINE:
1997
1998
1999    REPEAT
2000
2001       REPEAT
2002
2003          GETLINE( CHKPROMPT, TEXTLINE, LINELENGTH ):
2004          CURLINEPOS := 1:
2005          NEXTCHAR := TEXTLINE [1]:
2006
2007          ENDOFLINE := FALSE:
2008
2009       UNTIL (LINELENGTH <> NULLSTRLEN) AND NOT(INTERRUPT):
2010
2011
2012       ANSWER := UNKNOWN:
2013
2014       IF MATCH( YES )
2015          THEN
2016             ANSWER := YES:
2017
2018       IF MATCH( NO )
2019          THEN
2020             ANSWER := NO:
2021
2022       WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2023          GETCHAR:
2024
2025       IF NOT(ENDOFLINE)
2026          THEN
2027             ANSWER := UNKNOWN:
2028
2029       IF (ANSWER = UNKNOWN)
2030          THEN
```

```
2031              SHOWMESSAGE( ONLYNOYES ):
2032
2033     UNTIL (ANSWER IN [YES, NO]):
2034
2035     NEXTCHAR    := TEMPNEXTCHAR:
2036     TEXTLINE    := TEMPTEXTLINE:
2037     CURLINEPOS  := TEMPCURLINEPOS:
2038     LINELENGTH  := TEMPLINELENGTH:
2039     ENDOFLINE   := TEMPENDOFLINE;
2040
2041 END: (* GETRESPONSE *)
2042
2043 (*$Z *)
```

```
2044
2045 PROCEDURE DUMPREQUEST;
2046
2047 (*****************************************************************************)
2048 (*                                                                           *)
2049 (* DUMPREQUEST WILL DISPLAY ALL COMPONENTS OF THE CURRENT REQUEST.           *)
2050 (*                                                                           *)
2051 (* *** NOTE: THIS ROUTINE IS FOR DEBUG PURPOSES ONLY.                        *)
2052 (*                                                                           *)
2053 (*****************************************************************************)
2054
2055 VAR I : STRINGINDEX;
2056
2057 BEGIN (* DUMPREQUEST *)
2058
2059    WRITELN ('$$DUMP REQUEST? ');
2060    GETRESPONSE( ANSWER );
2061
2062    IF (ANSWER = YES)
2063       THEN
2064          WITH REQUEST DO
2065             BEGIN
2066
2067                WRITELN ('$$');
2068                WRITELN ('$$INPUT LINE = ');
2069                IF (LINELENGTH > 0)
2070                   THEN
2071                      FOR I := 1 TO LINELENGTH DO
2072                         WRITE (TEXTLINE [I]);
2073                WRITELN;
2074                WRITELN ('$$');
2075                WRITELN ('$$PARSED REQUEST');
2076                WRITE   ('$$NAME        = ');
2077                PRINT(NAME);
2078                WRITELN;
2079                WRITELN ('$$');
2080                WRITELN ('$$REPEATER    = ', REPEATER);
2081                WRITELN ('$$');
2082                WRITELN ('$$TEXTLEN     = ', TEXTLEN);
2083                IF (CURALPHABET = ASCII)
2084                   THEN
2085                      WRITE (PRINTASCII);
2086                WRITE   ('$$TEXTSTR     = ');
2087                IF (TEXTLEN > 0)
2088                   THEN
2089                      FOR I := 1 TO TEXTLEN DO
2090                         WRITE (TEXTSTR[I]);
2091                WRITELN;
2092                WRITELN ('$$');
2093                WRITELN ('$$NEWTEXTLEN = ', NEWTEXTLEN);
2094                IF (CURALPHABET = ASCII)
2095                   THEN
2096                      WRITE (PRINTASCII);
2097                WRITE   ('$$NEWTEXTSTR = ');
2098                IF (NEWTEXTLEN > 0)
2099                   THEN
2100                      FOR I := 1 TO NEWTEXTLEN DO
2101                         WRITE (NEWTEXTSTR[I]);
2102                WRITELN;
2103                WRITELN ('$$');
2104                WRITE   ('$$PLACE       = ');
2105                PRINT(PLACE);
2106                WRITELN;
```

```
2107                  WRITE    ('$$OPTION      = '):
2108                  PRINT(OPTION):
2109                  WRITELN:
2110                  WRITE    ('$$VALUE       = '):
2111                  PRINT(VALUE):
2112                  WRITELN:
2113                  WRITELN ('$$'):
2114                  WITH SPECIALSYM DO
2115                     WRITELN ('$$SPECIALSYM    ', 'LEN = ', LEN : 1,
2116                              ' VAL = ', VAL : 3):
2117                  WRITELN ('$$'):
2118                  WRITELN ('$$FILENAME    = ', FILENAME):
2119                  WRITELN ('$$'):
2120
2121               END: (* WITH *)
2122
2123     SHOWINPUT := TRUE:
2124
2125 END: (* DUMPREQUEST *)
2126
2127 (*$7 *)
```

```
2128
2129 PROCEDURE DUMPSTATUS;
2130
2131 (*************************************************************)
2132 (*                                                           *)
2133 (* DUMPSTATUS WILL DISPLAY CRITICAL VALUES OF HOPE STATUS VARIABLES.  *)
2134 (* IN ADDITION, THE INCORETEXT MAY BE DISPLAYED.             *)
2135 (*                                                           *)
2136 (* *** NOTE: THIS ROUTINE IS FOR DEBUG PURPOSES ONLY.        *)
2137 (*                                                           *)
2138 (*************************************************************)
2139
2140 VAR I : INCOREINDEX;
2141
2142 BEGIN (* DUMPSTATUS *)
2143
2144    WRITELN ('$$DUMP HOPE STATUS?');
2145    GETRESPONSE( ANSWER );
2146
2147    IF (ANSWER = YES)
2148       THEN
2149          BEGIN
2150             WRITELN ('$$');
2151             WRITELN ('$$ERROR = ', ERROR);
2152             WRITELN ('$$');
2153             WRITELN ('$$LINESFOUND = ', LINESFOUND);
2154             WRITELN ('$$');
2155             WRITE   ('$$FILESTATUS  = ');
2156             IF (FILESTATUS = OLD)
2157                THEN
2158                   WRITELN ('OLD' : 10)
2159                ELSE
2160                   WRITELN ('NEW' : 10);
2161             WRITELN ('$$FILECHANGES = ', FILECHANGES);
2162             WRITELN ('$$');
2163             WRITELN ('$$TOPSEQ      = ', TOPSEQ.FIRST, ' ',
2164                      TOPSEQ.LAST);
2165             WRITELN ('$$UPPERSEQ    = ', UPPERSEQ.FIRST, ' ',
2166                      UPPERSEQ.LAST);
2167             WRITELN ('$$HOLE        = ', HOLE.FIRST, ' ',
2168                      HOLE.LAST);
2169             WRITELN ('$$CURRENTLINE = ', CURRENTLINE.FIRST, ' ',
2170                      CURRENTLINE.LAST);
2171             WRITELN ('$$LOWERSEQ    = ', LOWERSEQ.FIRST, ' ',
2172                      LOWERSEQ.LAST);
2173             WRITELN ('$$BOTTOMSEQ   = ', BOTTOMSEQ.FIRST, ' ',
2174                      BOTTOMSEQ.LAST);
2175             WRITELN ('$$');
2176
2177             WRITELN ('$$DUMP INCORETEXT?');
2178             GETRESPONSE (ANSWER);
2179
2180             IF (ANSWER = YES)
2181                THEN
2182                   BEGIN
2183                      WRITELN ('$$');
2184                      WRITELN ('$$INCORETEXT = ');
2185                      WRITE   ('$$');
2186                      FOR I := 1 TO MAXNUMCHARS DO
2187                         IF (I < HOLE.FIRST) OR
2188                            (I > HOLE.LAST)
2189                            THEN
2190                               BEGIN
```

```
2191                                    WRITE (' ', INCORETEXT [I]):
2192                                    IF (I MOD 50 = 0)
2193                                     THEN
2194                                      BEGIN
2195                                       WRITELN:
2196                                       WRITE ('$'):
2197                                       END:
2198                                   END:
2199                          WRITELN:
2200                        END:
2201            END:
2202
2203     SHOWINPUT := TRUE:
2204
2205 END: (* JUMPSTATUS *)
2206
2207 (*$7 *)
```

```
2208
2209 PROCEDURE DUMPTOKENS;
2210
2211 (*************************************************************************)
2212 (*                                                                       *)
2213 (* DUMPTOKENS WILL DISPLAY THE PATTERN AND REPLACEMENT TOKENS.           *)
2214 (*                                                                       *)
2215 (* *** NOTE: THIS ROUTINE IS FOR DEBUG PURPOSES ONLY.                    *)
2216 (*                                                                       *)
2217 (*************************************************************************)
2218
2219 VAR I : STRINGINDEX;
2220
2221 BEGIN (* DUMPTOKENS *)
2222
2223    WRITELN ('$$DUMP TOKENS?');
2224    GETRESPONSE( ANSWER );
2225
2226    IF (ANSWER = YES)
2227       THEN
2228          BEGIN
2229             WRITELN ('$$');
2230             WRITELN ('$$PATTERN VALUES');
2231             WRITELN ('$$SYM CHR JOK  ELL.FIRST   ELL.LAST');
2232             WRITELN ('$$');
2233             IF (PATLEN = 0)
2234                THEN
2235                   WRITELN ('$$')
2236                ELSE
2237                   WITH PATTERN DO
2238                      FOR I := 1 TO PATLEN DO
2239                         BEGIN
2240                            WRITE ('$$');
2241                            CASE SYM [I] OF
2242                               WILDCARD   : WRITE ('W' : 3);
2243                               FREESTRING : WRITE ('F' : 3);
2244                               CONSTANT   : WRITE ('C' : 3);
2245                            END; (* CASE *)
2246                            WRITE (CHR [I] : 4);
2247                            WRITE (JOK [I] : 4);
2248                            IF (SYM [I] = FREESTRING)
2249                               THEN
2250                                  BEGIN
2251                                     WRITE (' ', ELL [I].FIRST);
2252                                     WRITE (' ', ELL [I].LAST);
2253                                  END;
2254                            WRITELN;
2255                         END;
2256             WRITELN ('$$');
2257             WRITELN ('$$REPLACEMENT VALUES');
2258             WRITELN ('$$SYM CHR JOK  ELL.FIRST   ELL.LAST');
2259             WRITELN ('$$');
2260             IF (REPLEN = 0)
2261                THEN
2262                   WRITELN ('$$')
2263                ELSE
2264                   WITH REPLACEMENT DO
2265                      FOR I := 1 TO REPLEN DO
2266                         BEGIN
2267                            WRITE ('$$');
2268                            CASE SYM [I] OF
2269                               WILDCARD   : WRITE ('W' : 3);
2270                               FREESTRING : WRITE ('F' : 3);
```

```
2271                                  CONSTANT   : WRITE ('C' : 3);
2272                                END; (* CASE *)
2273                                WRITE (CHR [I] : 4);
2274                                WRITE (JOK [I] : 4);
2275                                IF (SYM [I] = FREESTRING)
2276                                   THEN
2277                                    BEGIN
2278                                     WRITE (' ', ELL [I].FIRST);
2279                                     WRITE (' ', ELL [I].LAST);
2280                                    END;
2281                                WRITELN;
2282                             END;
2283                WRITELN ('$$');
2284             END;
2285
2286 END; (* DUMPTOKENS *)
2287
2288 (*$7 *)
```

```
2289
2290 FUNCTION GETNUMBER : BOOLEAN;
2291
2292 (*****************************************************************************)
2293 (*                                                                           *)
2294 (*  GETNUMBER ATTEMPTS TO MATCH A CHARACTER STRING REPRESENTATION OF         *)
2295 (*  A NUMBER. IF FOUND, THE NUMBER IS CONVERTED TO NUMERIC FORM AND          *)
2296 (*  PLACED INTO THE PARSED REQUEST. (ZERO IS NOT ALLOWED.)                   *)
2297 (*                                                                           *)
2298 (*****************************************************************************)
2299
2300 VAR STATE : FINITESTATE;
2301     EXIT  : BOOLEAN;
2302
2303     NUMBER : INTEGER;
2304
2305
2306 BEGIN (* GETNUMBER *)
2307
2308    GETNUMBER := FALSE;
2309    EXIT := FALSE;
2310    STATE := 1;
2311
2312    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2313       GETCHAR;
2314
2315    WHILE NOT(ERROR OR EXIT) DO
2316       CASE STATE OF
2317
2318          1 : IF (NEXTCHAR IN ['0'..'9'])
2319                THEN
2320                   BEGIN
2321                      GETNUMBER := TRUE;
2322                      NUMBER := ORD(NEXTCHAR) - ORD('0');
2323                      GETCHAR;
2324                      STATE := 2;
2325                   END
2326                ELSE
2327                   EXIT := TRUE;
2328
2329          2 : IF (NEXTCHAR IN ['0'..'9'])
2330                THEN
2331                   BEGIN
2332                      NUMBER := NUMBER*10 + ORD(NEXTCHAR) - ORD('0');
2333                      GETCHAR;
2334                      STATE := 2;
2335                   END
2336                ELSE
2337                   IF (NUMBER <> 0) AND
2338                      (NUMBER < MAXINT)
2339                      THEN
2340                         BEGIN
2341                            REQUEST.REPEATER := NUMBER;
2342                            EXIT := TRUE;
2343                         END
2344                      ELSE
2345                         SHOWERROR( ERRNUMBER );
2346
2347       END; (* CASE *)
2348
2349 END; (* GETNUMBER *)
2350
2351 (*97 *)
```

```
2352
2353 FUNCTION GETNAME : BOOLEAN;
2354
2355 (*************************************************************************)
2356 (*                                                                       *)
2357 (*  GETNAME ATTEMPTS TO MATCH AN IDENTIFIER FILENAME. THAT NAME          *)
2358 (*  MUST BEGIN WITH AN ALPHABETIC CHARACTER AND MAY CONTAIN ONLY         *)
2359 (*  ALPHA-NUMERIC CHARACTERS. IF THE IDENTIFIER IS FOUND, IT IS PUT      *)
2360 (*  INTO THE PARSED REQUEST.                                             *)
2361 (*                                                                       *)
2362 (*  *** NOTE :  UPPER/LOWER CASE ARE ALL CONVERTED TO UPPER CASE.        *)
2363 (*                                                                       *)
2364 (*************************************************************************)
2365
2366 VAR STATE : FINITESTATE;
2367     EXIT  : BOOLEAN;
2368     INDEX : 1..MAXFILENAMELEN;
2369
2370 BEGIN (* GETNAME *)
2371
2372    GETNAME := FALSE;
2373    EXIT := FALSE;
2374    STATE := 1;
2375
2376    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2377       GETCHAR;
2378
2379    WHILE NOT(ERROR OR EXIT) DO
2380       CASE STATE OF
2381
2382          1 : IF (NEXTCHAR IN ['A'..'Z'])
2383                 THEN
2384                    BEGIN
2385                       GETNAME := TRUE;
2386                       INDEX := 1;
2387                       REQUEST.FILENAME [INDEX] := NEXTCHAR;
2388                       GETCHAR;
2389                       STATE := 2;
2390                    END
2391                 ELSE
2392                    IF (CURLINEPOS < LINELENGTH)
2393                    THEN
2394                       BEGIN
2395                          IF (TEXTLINE [CURLINEPOS] = BREAKCODE) AND
2396                             (TEXTLINE [CURLINEPOS+1] IN ['A'..'Z'])
2397                             AND
2398                             (CURALPHABET = ASCII)
2399                             THEN
2400                              GETCHAR
2401                             ELSE
2402                              EXIT := TRUE;
2403                       END
2404                    ELSE
2405                       EXIT := TRUE;
2406
2407          2 : IF (NEXTCHAR IN ['A'..'Z','0'..'9']) AND
2408                 (INDEX < MAXFILENAMELEN)
2409                 THEN
2410                    BEGIN
2411                       INDEX := INDEX + 1;
2412                       REQUEST.FILENAME [INDEX] := NEXTCHAR;
2413                       GETCHAR;
2414                       STATE := 2;
```

```
2415                        END
2416                    ELSE
2417                        IF (CURLINEPOS < LINELENGTH)
2418                            THEN
2419                                BEGIN
2420                                    IF (TEXTLINE [CURLINEPOS] = BREAKCODE) AND
2421                                        (TEXTLINE [CURLINEPOS+1] IN
2422                                        ['A'..'Z']) AND
2423                                        (CURALPHABET = ASCII)
2424                                        THEN
2425                                         GETCHAR
2426                                        ELSE
2427                                         EXIT := TRUE;
2428                                END
2429                            ELSE
2430                                EXIT := TRUE;
2431
2432        END; (* CASE *)
2433
2434 END; (* GETNAME *)
2435
2436 (*$? *)
```

```
2437
2438 FUNCTION SPECIAL( CH : CHAR ) : BOOLEAN;
2439
2440 (*****************************************************************)
2441 (*                                                               *)
2442 (*  SPECIAL DETERMINES IF THE NEXT CHARACTER IS AN ELEMENT OF THE *)
2443 (*  IMPLEMENTATION DEPENDANT CHARACTERS ALLOWED AS A SPECIAL SYMBOL. *)
2444 (*                                                               *)
2445 (*****************************************************************)
2446
2447 BEGIN (* SPECIAL *)
2448
2449    SPECIAL := (CH = '\') OR (CH = '<') OR (CH = '>') OR
2450               (CH = '?') OR (CH = '&') OR (CH = '+') OR
2451               (CH = '*') OR (CH = '/') OR (CH = '''') OR
2452               (CH = '-') OR (CH = ',') OR (CH = '(') OR
2453               (CH = ')') OR (CH = '$') OR (CH = '=') OR
2454               (CH = '.') OR (CH = '#') OR (CH = '[') OR
2455               (CH = '%') OR (CH = '"') OR (CH = '_') OR
2456               (CH = ']') OR (CH = '^');
2457
2458 END; (* SPECIAL *)
2459
2460 (*$Z *)
```

```
2461
2462 FUNCTION GETSPECIALSYM : BOOLEAN;
2463
2464 (*************************************************************************)
2465 (*                                                                       *)
2466 (*  GETSPECIALSYM ATTEMPTS TO MATCH A STRING OF SPECIAL SYMBOLS. IF      *)
2467 (*  FOUND, THE CHARACTER STRING IS PLACED INTO THE PARSED REQUEST.       *)
2468 (*                                                                       *)
2469 (*************************************************************************)
2470
2471 VAR STATE : FINITESTATE;
2472     EXIT  : BOOLEAN;
2473
2474 BEGIN (* GETSPECIALSYM *)
2475
2476    GETSPECIALSYM := FALSE;
2477    EXIT := FALSE;
2478    STATE := 1;
2479
2480    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2481       GETCHAR;
2482
2483    WHILE NOT(ERROR OR EXIT) DO
2484       CASE STATE OF
2485
2486          1 : IF SPECIAL( NEXTCHAR )
2487                THEN
2488                   BEGIN
2489                      GETSPECIALSYM := TRUE;
2490                      REQUEST.SPECIALSYM.VAL [1] := NEXTCHAR;
2491                      REQUEST.SPECIALSYM.LEN := 1;
2492                      GETCHAR;
2493                      STATE := 2;
2494                   END
2495                ELSE
2496                   EXIT := TRUE;
2497
2498          2 : IF SPECIAL( NEXTCHAR )
2499                THEN
2500                   BEGIN
2501                      REQUEST.SPECIALSYM.VAL [2] := NEXTCHAR;
2502                      REQUEST.SPECIALSYM.LEN := 2;
2503                      GETCHAR;
2504                      STATE := 3;
2505                   END
2506                ELSE
2507                   EXIT := TRUE;
2508
2509          3 : IF SPECIAL( NEXTCHAR )
2510                THEN
2511                   BEGIN
2512                      REQUEST.SPECIALSYM.VAL [3] := NEXTCHAR;
2513                      REQUEST.SPECIALSYM.LEN := 3;
2514                      GETCHAR;
2515                      EXIT := TRUE;
2516                   END
2517                ELSE
2518                   EXIT := TRUE;
2519
2520       END; (* CASE *)
2521
2522 END; (* GETSPECIALSYM *)
2523
```

```
2524 (*$7 *)
2525
2526 FUNCTION GETENDREQUESTSYM : BOOLEAN;
2527
2528 (*************************************************************************)
2529 (*                                                                       *)
2530 (*  GETEND... INSURES THAT THERE ARE NO EXCESS CHARACTERS AT THE END     *)
2531 (*  OF THE REQUEST WHICH HAVE NOT BEEN RECOGNIZED (INDICATING A          *)
2532 (*  POSSIBLE SYNTAX ERROR.) THE ONLY NON-BLANK CHARACTER ALLOWED IS      *)
2533 (*  THE REQUEST SEPERATOR SYMBOL ELSE AN END OF LINE MUST BE FOUND.      *)
2534 (*                                                                       *)
2535 (*************************************************************************)
2536
2537 BEGIN (* GETENDREQUESTSYM *)
2538
2539    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2540       GETCHAR;
2541
2542    IF NEXTCHAR = ENDREQUESTSYM
2543       THEN
2544          BEGIN
2545             GETENDREQUESTSYM := TRUE;
2546             GETCHAR;
2547          END
2548       ELSE
2549          IF ENDOFLINE
2550             THEN
2551                GETENDREQUESTSYM := TRUE
2552             ELSE
2553                GETENDREQUESTSYM := FALSE;
2554
2555 END; (* GETENDREQUESTSYM *)
2556
2557 (*$7 *)
```

```
2558
2559 FUNCTION GETTEXTSTR : BOOLEAN;
2560
2561 (**************************************************************************)
2562 (*                                                                        *)
2563 (*  GETTEXTSTR ATTEMPTS TO MATCH FOR A TEXT STRING OF THE FORM:           *)
2564 (*                                                                        *)
2565 (*    =TEXT-STRING=  OR  =TEXT-STRING(NULL)  OR  =(NULL)                  *)
2566 (*                                                                        *)
2567 (*  WHERE THE SYMBOLS '=' ARE THE SAME SPECIAL CHARACTER DELIMITER        *)
2568 (*  (THE SECOND DELIMITER MAY BE OMITTED IF IT IS THE LAST CHARACTER      *)
2569 (*  ON THE LINE.) THE TEXT-STRING IS OPTIONAL (IF OMMITED, A NULL         *)
2570 (*  STRING IS ASSUMED.) THE TEXT-STRING AND ITS LENGTH ARE PLACED         *)
2571 (*  INTO THE PARSED REQUEST.                                              *)
2572 (*                                                                        *)
2573 (**************************************************************************)
2574
2575 VAR STATE : FINITESTATE;
2576     EXIT  : BOOLEAN;
2577     INDEX : 1..MAXSTRLEN;
2578     DELIMITER : CHAR;
2579
2580 BEGIN (* GETTEXTSTR *)
2581
2582    GETTEXTSTR := FALSE;
2583    EXIT := FALSE;
2584    STATE := 1;
2585
2586    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2587       GETCHAR;
2588
2589    WHILE NOT(ERROR OR EXIT) DO
2590       CASE STATE OF
2591
2592          1 : IF SPECIAL( NEXTCHAR )
2593              THEN
2594                 BEGIN
2595                    GETTEXTSTR := TRUE;
2596                    DELIMITER := NEXTCHAR;
2597                    GETCHAR;
2598                    STATE := 2;
2599                 END
2600              ELSE
2601                 EXIT := TRUE;
2602
2603          2 : IF (NEXTCHAR <> DELIMITER) AND NOT(ENDOFLINE)
2604              THEN
2605                 BEGIN
2606                    INDEX := 1;
2607                    REQUEST.TEXTSTR [INDEX] := NEXTCHAR;
2608                    REQUEST.TEXTLEN := INDEX;
2609                    GETCHAR;
2610                    STATE := 3;
2611                 END
2612              ELSE
2613                 BEGIN
2614                    REQUEST.TEXTSTR := NULLTEXT;
2615                    REQUEST.TEXTLEN := NULLSTRLEN;
2616                    STATE := 4;
2617                 END;
2618
2619          3 : IF (NEXTCHAR <> DELIMITER) AND NOT(ENDOFLINE)
2620              THEN
```

```
2621                        BEGIN
2622                           INDEX := INDEX + 1;
2623                           REQUEST.TEXTSTR [INDEX] := NEXTCHAR;
2624                           REQUEST.TEXTLEN := INDEX;
2625                           GETCHAR;
2626                           STATE := 3;
2627                        END
2628                     ELSE
2629                        STATE := 4;
2630
2631              4 : IF (NEXTCHAR = DELIMITER)
2632                     THEN
2633                        BEGIN
2634                           GETCHAR;
2635                           EXIT := TRUE;
2636                        END
2637                     ELSE
2638                        EXIT := TRUE;
2639
2640           END; (* CASE *)
2641
2642 END; (* GETTEXTSTR *)
2643
2644 (*$7 *)
```

```
2645
2646 FUNCTION GETOLDNEWTEXTSTR : BOOLEAN:
2647
2648 (*************************************************************************)
2649 (*                                                                       *)
2650 (*  GETOLD... ATTEMPTS TO MATCH FOR A NEW-TEXT-STRING OF THE FORM:       *)
2651 (*                                                                       *)
2652 (*      NEW-TEXT-STRING=  OR  NEW-TEXT-STRING(NULL)  OR  (NULL)          *)
2653 (*      =TEXT=NEW-TEXT=   OR  =TEXT=NEW-TEXT(NULL)   OR  =TEXT=(NULL)    *)
2654 (*                                                                       *)
2655 (*  WHERE THE CHARACTER '=' MUST MATCH THE FIRST DELIMITER FOUND BY       *)
2656 (*  GETTEXTSTRING. THE DELIMITER MAY BE OMMITTED IF IT IS THE LAST       *)
2657 (*  CHARACTER ON THE LINE. IF NO NEW-TEXT-STRING IS FOUND, A NULL        *)
2658 (*  TEXT IS ASSUMED. IF FOUND, THE NEW-TEXT-STRING AND ITS LENGTH ARE    *)
2659 (*  PLACED INTO THE PARSED REQUEST.                                      *)
2660 (*                                                                       *)
2661 (*************************************************************************)
2662
2663 VAR STATE : FINITESTATE:
2664     EXIT  : BOOLEAN:
2665     INDEX : 1..MAXSTRLEN:
2666     DELIMITER : CHAR:
2667
2668 BEGIN (* GETOLDNEWTEXTSTR *)
2669
2670    GETOLDNEWTEXTSTR := FALSE:
2671    EXIT := FALSE:
2672    STATE := 1:
2673
2674    WHILE (NEXTCHAR = BLANK) AND NOT(ENDOFLINE) DO
2675       GETCHAR:
2676
2677    WHILE NOT(ERROR OR EXIT) DO
2678       CASE STATE OF
2679
2680          1 : IF SPECIAL( NEXTCHAR )
2681                THEN
2682                   BEGIN
2683                      GETOLDNEWTEXTSTR := TRUE:
2684                      DELIMITER := NEXTCHAR:
2685                      GETCHAR:
2686                      STATE := 2:
2687                   END
2688                ELSE
2689                   EXIT := TRUE:
2690
2691          2 : IF (NEXTCHAR = DELIMITER)
2692                THEN
2693                   BEGIN
2694                      REQUEST.TEXTSTR := NULLTEXT:
2695                      REQUEST.TEXTLEN := NULLSTRLEN:
2696                      GETCHAR:
2697                      STATE := 4:
2698                   END
2699                ELSE
2700                   IF NOT(ENDOFLINE)
2701                      THEN
2702                         BEGIN
2703                            INDEX := 1:
2704                            REQUEST.TEXTSTR[INDEX] := NEXTCHAR:
2705                            REQUEST.TEXTLEN := INDEX:
2706                            GETCHAR:
2707                            STATE := 3:
```

```
2708                        END
2709                     ELSE
2710                        SHOWERROR( ERRMISSINGTEX ):
2711
2712          3 : IF (NEXTCHAR <> DELIMITER) AND NOT(ENDOFLINE)
2713              THEN
2714                 BEGIN
2715                    INDEX := INDEX + 1:
2716                    REQUEST.TEXTSTR [INDEX] := NEXTCHAR:
2717                    REQUEST.TEXTLEN := INDEX:
2718                    GETCHAR:
2719                    STATE := 3:
2720                 END
2721              ELSE
2722                 IF (NEXTCHAR = DELIMITER)
2723                 THEN
2724                    BEGIN
2725                       GETCHAR:
2726                       STATE := 4:
2727                    END
2728                 ELSE
2729                    SHOWERROR( ERRMISSINGTEXT ):
2730
2731          4 : IF (NEXTCHAR = DELIMITER) OR ENDOFLINE
2732              THEN
2733                 BEGIN
2734                    GETCHAR:
2735                    EXIT := TRUE:
2736                 END
2737              ELSE
2738                 IF NOT(ENDOFLINE)
2739                 THEN
2740                    BEGIN
2741                       INDEX := 1:
2742                       REQUEST.NEWTEXTSTR [INDEX] := NEXTCHAR:
2743                       REQUEST.NEWTEXTLEN := INDEX:
2744                       GETCHAR:
2745                       STATE := 5:
2746                    END:
2747
2748          5 : IF (NEXTCHAR = DELIMITER) OR ENDOFLINE
2749              THEN
2750                 BEGIN
2751                    GETCHAR:
2752                    EXIT := TRUE:
2753                 END
2754              ELSE
2755                 IF NOT(ENDOFLINE)
2756                 THEN
2757                    BEGIN
2758                       INDEX := INDEX + 1:
2759                       REQUEST.NEWTEXTSTR [INDEX] := NEXTCHAR:
2760                       REQUEST.NEWTEXTLEN := INDEX:
2761                       GETCHAR:
2762                       STATE := 5:
2763                    END:
2764
2765       END: (* CASE *)
2766
2767 END: (* GETOLDNEWTEXTSTR *)
2768
2769 (*$7 *)
```

```
2770
2771 FUNCTION GETREPEATER : BOOLEAN;
2772
2773 (*****************************************************************)
2774 (*                                                               *)
2775 (*  GETREPEATER ATTEMPTS TO MATCH A REPEATER IN THE FORM OF A NUMBER  *)
2776 (*  OR THE KEYWORD 'ALL'. IF 'ALL' IS FOUND, THE INTERNAL REPRESENT-  *)
2777 (*  ATION IS PLACED INTO THE PARSED REQUEST.                     *)
2778 (*                                                               *)
2779 (*****************************************************************)
2780
2781 BEGIN (* GETREPEATER *)
2782
2783    IF GETNUMBER
2784       THEN
2785          GETREPEATER := TRUE
2786       ELSE
2787          IF MATCH( ALL )
2788             THEN
2789                BEGIN
2790                   GETREPEATER := TRUE;
2791                   REQUEST.REPEATER := REPEATALL;
2792                END
2793             ELSE
2794                GETREPEATER := FALSE;
2795
2796 END; (* GETREPEATER *)
2797
2798 (*87 *)
```

```
2799
2800 FUNCTION GETFILE : BOOLEAN:
2801
2802 (*************************************************************************)
2803 (*                                                                       *)
2804 (*  GETFILE ATTEMPTS TO MATCH THE FILE OPTION :                          *)
2805 (*                                                                       *)
2806 (*        'FILE' 'IS' FILENAME                                           *)
2807 (*                                                                       *)
2808 (*  IF FOUND, THE OPTION IS PUT INTO THE PARSED REQUEST.                 *)
2809 (*                                                                       *)
2810 (*************************************************************************)
2811
2812 VAR STATE : FINITESTATE:
2813     EXIT  : BOOLEAN:
2814
2815 BEGIN (* GETFILE *)
2816
2817    GETFILE := FALSE:
2818    EXIT := FALSE:
2819    STATE := 1:
2820
2821    WHILE NOT(ERROR OR EXIT) DO
2822       CASE STATE OF
2823
2824          1 : IF MATCH( FILESYM )
2825                THEN
2826                   BEGIN
2827                      GETFILE := TRUE:
2828                      REQUEST.OPTION := FILESYM:
2829                      STATE := 2:
2830                   END
2831                ELSE
2832                   EXIT := TRUE:
2833
2834          2 : IF MATCH( ISSYM )
2835                THEN
2836                   STATE := 3
2837                ELSE
2838                   SHOWERROR( ERRISSYM ):
2839
2840          3 : IF GETNAME
2841                THEN
2842                   EXIT := TRUE
2843                ELSE
2844                   SHOWERROR( ERRFILENAME ):
2845
2846       END: (* CASE *)
2847
2848 END: (* GETFILE *)
2849
2850 (*$7 *)
```

```
2851
2852 FUNCTION GETMARGINELLIPSISJOKER : BOOLEAN;
2853
2854 (*************************************************************************)
2855 (*                                                                       *)
2856 (*  GETM...E...J... ATTEMPTS TO MATCH ONE OF THE OPTIONS :               *)
2857 (*                                                                       *)
2858 (*        'MARGIN'    'IS' 'NULL'/SPECIAL-CHARACTER                      *)
2859 (*                                                                       *)
2860 (*        'ELLIPSIS'  'IS' 'NULL'/SPECIAL-CHARACTER                      *)
2861 (*                                                                       *)
2862 (*        'JOKER'     'IS' 'NULL'/SPECIAL-CHARACTER                      *)
2863 (*                                                                       *)
2864 (*  IF FOUND, THE APPROPRIATE OPTION AND ITS VALUE ARE PLACED INTO       *)
2865 (*  THE PARSED REQUEST.                                                  *)
2866 (*                                                                       *)
2867 (*************************************************************************)
2868
2869 VAR STATE : FINITESTATE;
2870     EXIT  : BOOLEAN;
2871
2872 BEGIN (* GETMARGINELLIPSISJOKER *)
2873
2874    GETMARGINELLIPSISJOKER := FALSE;
2875    EXIT := FALSE;
2876    STATE := 1;
2877
2878    WHILE NOT(ERROR OR EXIT) DO
2879       CASE STATE OF
2880
2881          1 : IF MATCH( MARGIN )
2882                THEN
2883                   BEGIN
2884                      GETMARGINELLIPSISJOKER := TRUE;
2885                      REQUEST.OPTION := MARGIN;
2886                      STATE := 2;
2887                   END
2888                ELSE
2889                   IF MATCH( ELLIPSIS )
2890                      THEN
2891                         BEGIN
2892                            GETMARGINELLIPSISJOKER := TRUE;
2893                            REQUEST.OPTION := ELLIPSIS;
2894                            STATE := 2;
2895                         END
2896                      ELSE
2897                         IF MATCH( JOKER )
2898                            THEN
2899                               BEGIN
2900                                GETMARGINELLIPSISJOKER := TRUE;
2901                                REQUEST.OPTION := JOKER;
2902                                STATE := 2;
2903                               END
2904                            ELSE
2905                               EXIT := TRUE;
2906
2907          2 : IF MATCH( ISSYM )
2908                THEN
2909                   STATE := 3
2910                ELSE
2911                   SHOWERROR( ERRISSYM );
2912
2913          3 : IF GETSPECIALSYM
```

```
2914                    THEN
2915                       EXIT := TRUE
2916                    ELSE
2917                       IF MATCH( NULL )
2918                          THEN
2919                             BEGIN
2920                                REQUEST.SPECIALSYM := NULLSYM;
2921                                EXIT := TRUE;
2922                             END
2923                          ELSE
2924                             SHOWERROR( ERRSPECIALSYM );
2925
2926          END; (* CASE *)
2927
2928 END; (* GETMARGINELLIPSISJOKER *)
2929
2930 (*$Z *)
```

```
2931
2932 FUNCTION GETTOPBOTTOM : BOOLEAN;
2933
2934 (*************************************************************************)
2935 (*                                                                       *)
2936 (*  GETT...B... ATTEMPTS TO MATCH ONE OF THE OPTIONS :                   *)
2937 (*                                                                       *)
2938 (*        'TOP'    'IS' 'HERE'/'FIRST'                                   *)
2939 (*                                                                       *)
2940 (*        'BOTTOM' 'IS' 'HERE'/'LAST'                                    *)
2941 (*                                                                       *)
2942 (*  IF FOUND, THE APPROPRIATE OPTION AND ITS VALUE ARE PLACED INTO       *)
2943 (*  THE PARSED REQUEST.                                                  *)
2944 (*                                                                       *)
2945 (*************************************************************************)
2946
2947 VAR STATE : FINITESTATE;
2948     EXIT  : BOOLEAN;
2949
2950     OPTIONVAL : KEYSYMBOL;
2951
2952 BEGIN (* GETTOPBOTTOM *)
2953
2954    GETTOPBOTTOM := FALSE;
2955    EXIT := FALSE;
2956    STATE := 1;
2957
2958    WHILE NOT(ERROR OR EXIT) DO
2959       CASE STATE OF
2960
2961          1 : IF MATCH( TOP )
2962                THEN
2963                   BEGIN
2964                      GETTOPBOTTOM := TRUE;
2965                      REQUEST.OPTION := TOP;
2966                      OPTIONVAL := FIRST;
2967                      STATE := 2;
2968                   END
2969                ELSE
2970                   IF MATCH( BOTTOM )
2971                      THEN
2972                         BEGIN
2973                            GETTOPBOTTOM := TRUE;
2974                            REQUEST.OPTION := BOTTOM;
2975                            OPTIONVAL := LAST;
2976                            STATE := 2;
2977                         END
2978                      ELSE
2979                         EXIT := TRUE;
2980
2981          2 : IF MATCH( ISSYM )
2982                THEN
2983                   STATE := 3
2984                ELSE
2985                   SHOWERROR( ERRISSYM );
2986
2987          3 : IF MATCH( HERE )
2988                THEN
2989                   BEGIN
2990                      REQUEST.VALUE := HERE;
2991                      EXIT := TRUE;
2992                   END
2993                ELSE
```

```
2994                        IF MATCH( OPTIONVAL )
2995                           THEN
2996                              BEGIN
2997                                 REQUEST.VALUE := OPTIONVAL;
2998                                 EXIT := TRUE;
2999                              END
3000                           ELSE
3001                              IF (OPTIONVAL = FIRST)
3002                              THEN
3003                                 SHOWERROR( ERRFIRST )
3004                              ELSE
3005                                 SHOWERROR( ERRLAST );
3006
3007        END; (* CASE *)
3008
3009 END; (* GETTOPBOTTOM *)
3010
3011 (*$Z *)
```

```
3012
3013 FUNCTION GETALPHABET : BOOLEAN;
3014
3015 (*************************************************************************)
3016 (*                                                                       *)
3017 (*  GETALPHABET ATTEMPTS TO MATCH THE OPTION :                           *)
3018 (*                                                                       *)
3019 (*        'ALPHABET' 'IS' 'NORMAL'/'ASCII'                               *)
3020 (*                                                                       *)
3021 (*  IF FOUND, THE OPTION AND VALUE IS PUT INTO THE PARSED REQUEST.       *)
3022 (*                                                                       *)
3023 (*************************************************************************)
3024
3025 VAR STATE : FINITESTATE;
3026     EXIT  : BOOLEAN;
3027
3028 BEGIN (* GETALPHABET *)
3029
3030    GETALPHABET := FALSE;
3031    EXIT := FALSE;
3032    STATE := 1;
3033
3034    WHILE NOT(ERROR OR EXIT) DO
3035       CASE STATE OF
3036
3037          1 : IF MATCH( ALPHABET )
3038                THEN
3039                   BEGIN
3040                      GETALPHABET := TRUE;
3041                      REQUEST.OPTION := ALPHABET;
3042                      STATE := 2;
3043                   END
3044                ELSE
3045                   EXIT := TRUE;
3046
3047          2 : IF MATCH( ISSYM )
3048                THEN
3049                   STATE := 3
3050                ELSE
3051                   SHOWERROR( ERRISSYM );
3052
3053          3 : IF MATCH( NORMAL )
3054                THEN
3055                   BEGIN
3056                      REQUEST.VALUE := NORMAL;
3057                      EXIT := TRUE;
3058                   END
3059                ELSE
3060                   IF MATCH( ASCII )
3061                   THEN
3062                      BEGIN
3063                         REQUEST.VALUE := ASCII;
3064                         EXIT := TRUE;
3065                      END
3066                   ELSE
3067                      SHOWERROR( ERRNORMAL );
3068
3069       END; (* CASE *)
3070
3071 END; (* GETALPHABET *)
3072
3073 (*17 *)
```

```
3074
3075 FUNCTION GETFORWARDBACKWARDLISTDELETE : BOOLEAN;
3076
3077 (*****************************************************************)
3078 (*                                                               *)
3079 (*  GETF...B...L...D... ATTEMPTS TO MATCH ONE OF THE FORMS :     *)
3080 (*                                                               *)
3081 (*       'FORWARD'   (REPEATER) (TEXT-STRING)                    *)
3082 (*                                                               *)
3083 (*       'BACKWARD'  (REPEATER) (TEXT-STRING)                    *)
3084 (*                                                               *)
3085 (*       'LIST'      (REPEATER) (TEXT-STRING)                    *)
3086 (*                                                               *)
3087 (*       'DELETE'    (REPEATER) (TEXT-STRING)                    *)
3088 (*                                                               *)
3089 (*  WHERE REPEATER AND TEXT-STRING ARE OPTIONAL. IF REPEATER IS   *)
3090 (*  OMITTED, THE DEFAULT REPEATER (1) IS INSERTED. IF TEXT-STRING *)
3091 (*  IS OMITTED, A NULL TEXT STRING IS INSERTED. IF FOUND, THE NAME *)
3092 (*  IS INSERTED INTO THE PARSED REQUEST.                         *)
3093 (*                                                               *)
3094 (*****************************************************************)
3095
3096 VAR STATE : FINITESTATE;
3097     EXIT  : BOOLEAN;
3098
3099 BEGIN (* GETFORWARDBACKWARDLISTDELETE *)
3100
3101    GETFORWARDBACKWARDLISTDELETE := FALSE;
3102    EXIT := FALSE;
3103    STATE := 1;
3104
3105    WHILE NOT(ERROR OR EXIT) DO
3106       CASE STATE OF
3107
3108          1 : IF MATCH( FORWARD )
3109              THEN
3110                 BEGIN
3111                    GETFORWARDBACKWARDLISTDELETE := TRUE;
3112                    REQUEST.NAME := FORWARD;
3113                    STATE := 2;
3114                 END
3115              ELSE
3116                 IF MATCH( BACKWARD )
3117                 THEN
3118                    BEGIN
3119                       GETFORWARDBACKWARDLISTDELETE := TRUE;
3120                       REQUEST.NAME := BACKWARD;
3121                       STATE := 2;
3122                    END
3123                 ELSE
3124                    IF MATCH( LIST )
3125                    THEN
3126                       BEGIN
3127                        GETFORWARDBACKWARDLISTDELETE := TRUE;
3128                        REQUEST.NAME := LIST;
3129                        STATE := 2;
3130                       END
3131                    ELSE
3132                       IF MATCH( DELETE )
3133                       THEN
3134                        BEGIN
3135                         GETFORWARDBACKWARDLISTDELETE := TRUE;
3136                         REQUEST.NAME := DELETE;
```

```
3137                                        STATE := 2:
3138                                       END
3139                                      ELSE
3140                                       EXIT := TRUE:
3141
3142            2 : IF GETREPEATER
3143                THEN
3144                   STATE := 3
3145                ELSE
3146                   BEGIN
3147                      REQUEST.REPEATER := DEFAULTRPTR:
3148                      STATE := 3:
3149                   END:
3150
3151            3 : IF GETTEXTSTR
3152                THEN
3153                   EXIT := TRUE
3154                ELSE
3155                   BEGIN
3156                      REQUEST.TEXTSTR := NULLTEXT:
3157                      REQUEST.TEXTLEN := NULLSTRLEN:
3158                      EXIT := TRUE:
3159                   END:
3160
3161        END: (* CASE *)
3162
3163 END: (* GETFORWARDBACKWARDLISTDELETE *)
3164
3165 (*$Z *)
```

```
3166
3167 FUNCTION GETTRANSFERCOPY : BOOLEAN;
3168
3169 (*************************************************************************)
3170 (*                                                                       *)
3171 (*  GETT...C... ATTEMPTS TO MATCH ONE OF THE FORMS :                     *)
3172 (*                                                                       *)
3173 (*       'TRANSFER' (REPEATER) (TEXT-STRING) 'INTO' FILENAME             *)
3174 (*                                                                       *)
3175 (*       'COPY'     (REPEATER) (TEXT-STRING) 'INTO' FILENAME             *)
3176 (*                                                                       *)
3177 (*  WHERE REPEATER AND TEXT-STRING ARE OPTIONAL. IF REPEATER IS           *)
3178 (*  OMITTED, THE DEFAULT REPEATER (1) IS INSERTED. IF TEXT-STRING        *)
3179 (*  IS OMMITTED, A NULL TEXT STRING IS INSERTED. IF FOUND, THE NAME      *)
3180 (*  OF THE REQUEST IS PLACED INTO THE PARSED REQUEST.                    *)
3181 (*                                                                       *)
3182 (*************************************************************************)
3183
3184 VAR STATE : FINITESTATE;
3185     EXIT  : BOOLEAN;
3186
3187 BEGIN (* GETTRANSFERCOPY *)
3188
3189    GETTRANSFERCOPY := FALSE;
3190    EXIT := FALSE;
3191    STATE := 1;
3192
3193    WHILE NOT(ERROR OR EXIT) DO
3194       CASE STATE OF
3195
3196          1 : IF MATCH( TRANSFER )
3197                THEN
3198                   BEGIN
3199                      GETTRANSFERCOPY := TRUE;
3200                      REQUEST.NAME := TRANSFER;
3201                      STATE := 2;
3202                   END
3203                ELSE
3204                   IF MATCH( COPY )
3205                      THEN
3206                         BEGIN
3207                            GETTRANSFERCOPY := TRUE;
3208                            REQUEST.NAME := COPY;
3209                            STATE := 2;
3210                         END
3211                      ELSE
3212                         EXIT := TRUE;
3213
3214          2 : IF GETREPEATER
3215                THEN
3216                   STATE := 3
3217                ELSE
3218                   BEGIN
3219                      REQUEST.REPEATER := DEFAULTRPTR;
3220                      STATE := 3;
3221                   END;
3222
3223          3 : IF GETTEXTSTR
3224                THEN
3225                   STATE := 4
3226                ELSE
3227                   BEGIN
3228                      REQUEST.TEXTSTR := NULLTEXT;
```

```
3229                        REQUEST.TEXTLEN := NULLSTRLEN:
3230                        STATE := 4:
3231                     END:
3232
3233           4 : IF MATCH( INTO )
3234                  THEN
3235                     STATE := 5
3236                  ELSE
3237                     SHOWERROR( ERRINTO ):
3238
3239           5 : IF GETNAME
3240                  THEN
3241                     EXIT := TRUE
3242                  ELSE
3243                     SHOWERROR( ERRFILENAME ):
3244
3245        END: (* CASE *)
3246
3247 END: (* GETTRANSFERCOPY *)
3248
3249 (*$7 *)
```

```
3250
3251 FUNCTION GETMAKE : BOOLEAN;
3252
3253 (*************************************************************)
3254 (*                                                           *)
3255 (*  GETMAKE ATTEMPTS TO MATCH THE FOLLOWING FORM:            *)
3256 (*                                                           *)
3257 (*       'MAKE' (REPEATER) TEXT-STRING NEW-TEXT-STRING       *)
3258 (*                                                           *)
3259 (*  IF FOUND, THE NAME IS PLACED INTO THE PARSED REQUEST.    *)
3260 (*                                                           *)
3261 (*************************************************************)
3262
3263 VAR STATE : FINITESTATE;
3264     EXIT  : BOOLEAN;
3265
3266 BEGIN (* GETMAKE *)
3267
3268    GETMAKE := FALSE;
3269    EXIT := FALSE;
3270    STATE := 1;
3271
3272    WHILE NOT(ERROR OR EXIT) DO
3273       CASE STATE OF
3274
3275          1 : IF MATCH( MAKE )
3276                THEN
3277                   BEGIN
3278                      GETMAKE := TRUE;
3279                      REQUEST.NAME := MAKE;
3280                      STATE := 2;
3281                   END
3282                ELSE
3283                   EXIT := TRUE;
3284
3285          2 : IF GETREPEATER
3286                THEN
3287                   STATE := 3
3288                ELSE
3289                   BEGIN
3290                      REQUEST.REPEATER := DEFAULTRPTR;
3291                      STATE := 3;
3292                   END;
3293
3294          3 : IF GETOLDNEWTEXTSTR
3295                THEN
3296                   EXIT := TRUE
3297                ELSE
3298                   SHOWERROR( ERRMISSINGTEXT );
3299
3300       END; (* CASE *)
3301
3302 END; (* GETMAKE *)
3303
3304 (*$7 *)
```

```
3305
3306 FUNCTION GETINSERT : BOOLEAN;
3307
3308 (*************************************************************************)
3309 (*                                                                       *)
3310 (*  GETINSERT ATTEMPTS TO MATCH THE FOLLOWING FORM:                      *)
3311 (*                                                                       *)
3312 (*       'INSERT' (TEXT-STRING/'FILE' FILENAME) (AFTER/BEFORE/OVER)      *)
3313 (*                                                                       *)
3314 (*  IF TEXT-STRING OR FILENAME ARE OMMITED, INPUT FROM THE TERMINAL      *)
3315 (*  IS ASSUMED. IF AFTER, BEFORE, OR OVER ARE OMITTED, AFTER IS          *)
3316 (*  ASSUMED. IF FOUND, THE NAME AND PLACE ARE PUT INTO THE PARSED        *)
3317 (*  REQUEST.                                                             *)
3318 (*                                                                       *)
3319 (*************************************************************************)
3320
3321 VAR STATE : FINITESTATE;
3322     EXIT  : BOOLEAN;
3323
3324 BEGIN (* GETINSERT *)
3325
3326    GETINSERT := FALSE;
3327    EXIT := FALSE;
3328    STATE := 1;
3329
3330    WHILE NOT(ERROR OR EXIT) DO
3331       CASE STATE OF
3332
3333          1 : IF MATCH( INSERT )
3334              THEN
3335                 BEGIN
3336                    GETINSERT := TRUE;
3337                    REQUEST.NAME := INSERT;
3338                    STATE := 2;
3339                 END
3340              ELSE
3341                 EXIT := TRUE;
3342
3343          2 : IF GETTEXTSTR
3344              THEN
3345                 STATE := 3
3346              ELSE
3347                 IF MATCH( FILESYM )
3348                 THEN
3349                    BEGIN
3350                       IF GETNAME
3351                          THEN
3352                           STATE := 3
3353                          ELSE
3354                           SHOWERROR( ERRFILENAME );
3355                    END
3356                 ELSE
3357                    BEGIN
3358                       REQUEST.FILENAME := INPUTFILE;
3359                       STATE := 3;
3360                    END;
3361
3362          3 : IF MATCH( AFTER )
3363              THEN
3364                 BEGIN
3365                    REQUEST.PLACE := AFTER;
3366                    EXIT := TRUE;
3367                 END
```

```
3368                    ELSE
3369                       IF MATCH( OVER )
3370                          THEN
3371                             BEGIN
3372                                REQUEST.PLACE := OVER;
3373                                EXIT := TRUE;
3374                             END
3375                          ELSE
3376                             IF MATCH( BEFORE )
3377                                THEN
3378                                   BEGIN
3379                                    REQUEST.PLACE := BEFORE;
3380                                    EXIT := TRUE;
3381                                   END
3382                                ELSE
3383                                   BEGIN
3384                                    REQUEST.PLACE := AFTER;
3385                                    EXIT := TRUE;
3386                                   END;
3387
3388          END; (* CASE *)
3389
3390 END; (* GETINSERT *)
3391
3392 (*$7 *)
```

```
3393
3394 FUNCTION GETSHOW : BOOLEAN;
3395
3396 (*************************************************************************)
3397 (*                                                                       *)
3398 (*  GETSHOW ATTEMPTS TO MATCH ONE OF THE FOLLOWING FORMS:                *)
3399 (*                                                                       *)
3400 (*        'SHOW'  ASSUMPTION                                             *)
3401 (*                                                                       *)
3402 (*        'SHOW'                                                         *)
3403 (*                                                                       *)
3404 (*  WHERE ASSUMPTION IS THE NAME OF A SPECIFIC ASSUMPTION. IF FOUND,     *)
3405 (*  THE NAME (AND ASSUMPTION) IS PLACED INTO THE PARSED REQUEST.         *)
3406 (*                                                                       *)
3407 (*************************************************************************)
3408
3409 VAR STATE : FINITESTATE;
3410     EXIT  : BOOLEAN;
3411     PATH : 1..8;
3412     FOUND : BOOLEAN;
3413
3414 BEGIN (* GETSHOW *)
3415
3416    GETSHOW := FALSE;
3417    EXIT := FALSE;
3418    STATE := 1;
3419    PATH := 1;
3420    FOUND := FALSE;
3421
3422    WHILE NOT(ERROR OR EXIT) DO
3423       CASE STATE OF
3424
3425          1 : IF MATCH( SHOW )
3426                 THEN
3427                    BEGIN
3428                       GETSHOW := TRUE;
3429                       REQUEST.NAME := SHOW;
3430                       STATE := 2;
3431                    END
3432                 ELSE
3433                    EXIT := TRUE;
3434
3435          2 : WHILE NOT(ERROR OR EXIT) DO
3436                 CASE PATH OF
3437
3438                    1 : IF MATCH( FILESYM )
3439                           THEN
3440                              BEGIN
3441                                 REQUEST.OPTION := FILESYM;
3442                                 EXIT := TRUE;
3443                              END
3444                           ELSE
3445                              PATH := 2;
3446
3447                    2 : IF MATCH( MARGIN )
3448                           THEN
3449                              BEGIN
3450                                 REQUEST.OPTION := MARGIN;
3451                                 EXIT := TRUE;
3452                              END
3453                           ELSE
3454                              PATH := 3;
3455
```

```
3456                        3 : IF MATCH( ELLIPSIS )
3457                            THEN
3458                               BEGIN
3459                                  REQUEST.OPTION := ELLIPSIS;
3460                                  EXIT := TRUE;
3461                               END
3462                            ELSE
3463                               PATH := 4;
3464
3465                        4 : IF MATCH( JOKER )
3466                            THEN
3467                               BEGIN
3468                                  REQUEST.OPTION := JOKER;
3469                                  EXIT := TRUE;
3470                               END
3471                            ELSE
3472                               PATH := 5;
3473
3474                        5 : IF MATCH( TOP )
3475                            THEN
3476                               BEGIN
3477                                  REQUEST.OPTION := TOP;
3478                                  EXIT := TRUE;
3479                               END
3480                            ELSE
3481                               PATH := 6;
3482
3483                        6 : IF MATCH( BOTTOM )
3484                            THEN
3485                               BEGIN
3486                                  REQUEST.OPTION := BOTTOM;
3487                                  EXIT := TRUE;
3488                               END
3489                            ELSE
3490                               PATH := 7;
3491
3492                        7 : IF MATCH( ALPHABET )
3493                            THEN
3494                               BEGIN
3495                                  REQUEST.OPTION := ALPHABET;
3496                                  EXIT := TRUE;
3497                               END
3498                            ELSE
3499                               PATH := 8;
3500
3501                        8 : IF MATCH( CATALOG )
3502                            THEN
3503                               BEGIN
3504                                  REQUEST.OPTION := CATALOG;
3505                                  EXIT := TRUE;
3506                               END
3507                            ELSE
3508                               BEGIN
3509                                  REQUEST.OPTION := ALL;
3510                                  EXIT := TRUE;
3511                               END;
3512
3513                     END; (* CASE PATH *)
3514
3515           END; (* CASE STATE *)
3516
3517 END; (* GETSHOW *)
3518
3519 (*$Z *)
```

```
3520
3521 FUNCTION GETPRESERVE : BOOLEAN;
3522
3523 (*************************************************************************)
3524 (*                                                                       *)
3525 (*  GETPRESERVE ATTEMPTS TO MATCH THE FOLLOWING FORM:                    *)
3526 (*                                                                       *)
3527 (*        'PRESERVE' (FILENAME)                                          *)
3528 (*                                                                       *)
3529 (*  IF FILENAME IS OMMITTED, THE CURRENT FILE IS ASSUMED.                *)
3530 (*                                                                       *)
3531 (*************************************************************************)
3532
3533 VAR STATE : FINITESTATE;
3534     EXIT  : BOOLEAN;
3535
3536 BEGIN (* GETPRESERVE *)
3537
3538    GETPRESERVE := FALSE;
3539    EXIT := FALSE;
3540    STATE := 1;
3541
3542    WHILE NOT(ERROR OR EXIT) DO
3543       CASE STATE OF
3544
3545          1 : IF MATCH( PRESERVE )
3546                THEN
3547                   BEGIN
3548                      GETPRESERVE := TRUE;
3549                      REQUEST.NAME := PRESERVE;
3550                      STATE := 2;
3551                   END
3552                ELSE
3553                   EXIT := TRUE;
3554
3555          2 : IF GETNAME
3556                THEN
3557                   EXIT := TRUE
3558                ELSE
3559                   BEGIN
3560                      REQUEST.FILENAME := CURFILENAME;
3561                      EXIT := TRUE;
3562                   END;
3563
3564       END; (* CASE *)
3565
3566 END; (* GETPRESERVE *)
3567
3568 (*$7 *)
```

```
3569
3570 FUNCTION GETDESTROY : BOOLEAN;
3571
3572 (*****************************************************************************)
3573 (*                                                                           *)
3574 (*  GETDESTROY ATTEMPTS TO MATCH THE FOLLOWING FORM:                         *)
3575 (*                                                                           *)
3576 (*        'DESTROY' (FILENAME)                                               *)
3577 (*                                                                           *)
3578 (*  IF FILENAME IS OMMITTED, THE CURRENT FILE IS ASSUMED.                    *)
3579 (*                                                                           *)
3580 (*****************************************************************************)
3581
3582
3583 VAR STATE : FINITESTATE;
3584     EXIT  : BOOLEAN;
3585
3586 BEGIN (* GETDESTROY *)
3587
3588    GETDESTROY := FALSE;
3589    EXIT := FALSE;
3590    STATE := 1;
3591
3592    WHILE NOT(ERROR OR EXIT) DO
3593       CASE STATE OF
3594
3595          1 : IF MATCH( DESTROY )
3596                THEN
3597                   BEGIN
3598                      GETDESTROY := TRUE;
3599                      REQUEST.NAME := DESTROY;
3600                      STATE := 2;
3601                   END
3602                ELSE
3603                   EXIT := TRUE;
3604
3605          2 : IF GETNAME
3606                THEN
3607                   EXIT := TRUE
3608                ELSE
3609                   BEGIN
3610                      REQUEST.FILENAME := CURFILENAME;
3611                      EXIT := TRUE;
3612                   END;
3613
3614       END; (* CASE *)
3615
3616 END; (* GETDESTROY *)
3617
3618 (*$Z *)
```

```
3619
3620 FUNCTION GETASSUME : BOOLEAN;
3621
3622 (*************************************************************************)
3623 (*                                                                       *)
3624 (*  GETASSUME ATTEMPTS TO MATCH THE FOLLOWING FORM:                      *)
3625 (*                                                                       *)
3626 (*        'ASSUME' OPTION                                                *)
3627 (*                                                                       *)
3628 (*  WHERE OPTION IS ONE OF THE LEGAL ASSUME OPTIONS (FILE, MARGIN,       *)
3629 (*  ELLIPSIS, JOKER, TOP, BOTTOM, ALPHABET.) IF FOUND, THE NAME IS       *)
3630 (*  PLACED INTO THE PARSED REQUEST.                                      *)
3631 (*                                                                       *)
3632 (*************************************************************************)
3633
3634 VAR STATE : FINITESTATE;
3635     EXIT  : BOOLEAN;
3636     PATH  : 1..4;
3637
3638 BEGIN (* GETASSUME *)
3639
3640    GETASSUME := FALSE;
3641    EXIT := FALSE;
3642    STATE := 1;
3643    PATH := 1;
3644
3645    WHILE NOT(ERROR OR EXIT) DO
3646       CASE STATE OF
3647
3648          1 : IF MATCH( ASSUME )
3649                 THEN
3650                    BEGIN
3651                       GETASSUME := TRUE;
3652                       REQUEST.NAME := ASSUME;
3653                       STATE := 2;
3654                    END
3655                 ELSE
3656                    EXIT := TRUE;
3657
3658          2 : WHILE NOT(EPPOP OR EXIT) DO
3659                 CASE PATH OF
3660
3661                    1 : IF GETFILE
3662                           THEN
3663                              EXIT := TRUE
3664                           ELSE
3665                              PATH := 2;
3666
3667                    2 : IF GETMARGINELLIPSISJOKER
3668                           THEN
3669                              EXIT := TRUE
3670                           ELSE
3671                              PATH := 3;
3672
3673                    3 : IF GETTOPBOTTOM
3674                           THEN
3675                              EXIT := TRUE
3676                           ELSE
3677                              PATH := 4;
3678
3679                    4 : IF GETALPHABET
3680                           THEN
3681                              EXIT := TRUE
```

```
3682                                    ELSE
3683                                       SHOWERROR( EPROPTION ) ;
3684
3685                          END; (* CASE PATH *)
3686
3687             END; (* CASE STATE *)
3688
3689 END; (* GETASSUME *)
3690
3691 (*$Z *)
```

```
3692
3693
3694
3695 FUNCTION GETUNDOQUIT : BOOLEAN;
3696
3697 (*************************************************************************)
3698 (*                                                                       *)
3699 (*  GETU...Q... ATTEMPTS TO MATCH ONE OF THE FORMS :                     *)
3700 (*                                                                       *)
3701 (*        'UNDO'  OR  'QUIT'                                             *)
3702 (*                                                                       *)
3703 (*  IF FOUND, THE APPROPRIATE NAME IS PUT INTO THE PARSED REQUEST.       *)
3704 (*                                                                       *)
3705 (*************************************************************************)
3706
3707 BEGIN (* GETUNDOQUIT *)
3708
3709    IF MATCH( UNDO )
3710       THEN
3711          BEGIN
3712             REQUEST.NAME := UNDO;
3713             GETUNDOQUIT := TRUE;
3714          END
3715       ELSE
3716          IF MATCH( QUIT )
3717             THEN
3718                BEGIN
3719                   REQUEST.NAME := QUIT;
3720                   GETUNDOQUIT := TRUE;
3721                END
3722             ELSE
3723                GETUNDOQUIT := FALSE;
3724
3725 END; (* GETUNDOQUIT *)
3726
3727 (*$7 *)
```

```
3728
3729 FUNCTION GETGRIPE : BOOLEAN:
3730
3731 (*************************************************************************)
3732 (*                                                                       *)
3733 (*  GETGRIPE ATTEMPTS TO MATCH THE FOLLOWING FORM:                       *)
3734 (*                                                                       *)
3735 (*        'GRIPE'                                                        *)
3736 (*                                                                       *)
3737 (*  IF FOUND, THE NAME IS PLACED INTO THE PARSED REQUEST.                *)
3738 (*                                                                       *)
3739 (*************************************************************************)
3740
3741 BEGIN (* GETGRIPE *)
3742
3743    IF MATCH( GRIPE )
3744       THEN
3745          BEGIN
3746             REQUEST.NAME := GRIPE:
3747             GETGRIPE := TRUE:
3748          END
3749       ELSE
3750          GETGRIPE := FALSE:
3751
3752 END: (* GETGRIPE *)
3753
3754 (*$7 *)
```

```
3755
3756 FUNCTION GETREQUEST : BOOLEAN:
3757
3758 (*************************************************************************)
3759 (*                                                                       *)
3760 (*  GETREQUEST ATTEMPTS TO FIND A LEGAL EDITTING REQUEST. IF ONE IS      *)
3761 (*  FOUND, A CHECK IS MADE FOR EXCESS CHARACTERS (INDICATING A           *)
3762 (*  POSSIBLE SYNTAX ERROR.)                                              *)
3763 (*                                                                       *)
3764 (*************************************************************************)
3765
3766 VAR STATE : FINITESTATE:
3767     EXIT  : BOOLEAN:
3768     PATH  : 1..10:
3769     FOUND : BOOLEAN:
3770
3771 BEGIN (* GETREQUEST *)
3772
3773    GETREQUEST := FALSE:
3774    EXIT := FALSE:
3775    STATE := 1:
3776    PATH := 1:
3777    FOUND := FALSE:
3778
3779    WHILE NOT(ERROR OR EXIT) DO
3780       CASE STATE OF
3781
3782          1 : WHILE NOT(FOUND) AND NOT(ERROR OR EXIT) DO
3783                 CASE PATH OF
3784
3785                    1 : IF GETPRESERVE
3786                        THEN
3787                           BEGIN
3788                              GETREQUEST := TRUE:
3789                              FOUND := TRUE:
3790                              STATE := 2:
3791                           END
3792                        ELSE
3793                           PATH := 2:
3794
3795                    2 : IF GETDESTROY
3796                        THEN
3797                           BEGIN
3798                              GETREQUEST := TRUE:
3799                              FOUND := TRUE:
3800                              STATE := 2:
3801                           END
3802                        ELSE
3803                           PATH := 3:
3804
3805                    3 : IF GETFORWARDBACKWARDLISTDELETE
3806                        THEN
3807                           BEGIN
3808                              GETREQUEST := TRUE:
3809                              FOUND := TRUE:
3810                              STATE := 2:
3811                           END
3812                        ELSE
3813                           PATH := 4:
3814
3815                    4 : IF GETTRANSFERCOPY
3816                        THEN
3817                           BEGIN
```

```
3818                        GETREQUEST := TRUE:
3819                        FOUND := TRUE:
3820                        STATE := 2:
3821                     END
3822                  ELSE
3823                     PATH := 5:
3824
3825            5 : IF GETINSERT
3826                  THEN
3827                     BEGIN
3828                        GETREQUEST := TRUE:
3829                        FOUND := TRUE:
3830                        STATE := 2:
3831                     END
3832                  ELSE
3833                     PATH := 6:
3834
3835            6 : IF GETMAKE
3836                  THEN
3837                     BEGIN
3838                        GETREQUEST := TRUE:
3839                        FOUND := TRUE:
3840                        STATE := 2:
3841                     END
3842                  ELSE
3843                     PATH := 7:
3844
3845            7 : IF GETASSUME
3846                  THEN
3847                     BEGIN
3848                        GETREQUEST := TRUE:
3849                        FOUND := TRUE:
3850                        STATE := 2:
3851                     END
3852                  ELSE
3853                     PATH := 8:
3854
3855            8 : IF GETSHOW
3856                  THEN
3857                     BEGIN
3858                        GETREQUEST := TRUE:
3859                        FOUND := TRUE:
3860                        STATE := 2:
3861                     END
3862                  ELSE
3863                     PATH := 9:
3864
3865            9 : IF GETUNDOQUIT
3866                  THEN
3867                     BEGIN
3868                        GETREQUEST := TRUE:
3869                        FOUND := TRUE:
3870                        STATE := 2:
3871                     END
3872                  ELSE
3873                     PATH := 10:
3874
3875           10 : IF GETGRIPE
3876                  THEN
3877                     BEGIN
3878                      GETREQUEST := TRUE:
3879                      FOUND := TRUE:
3880                      STATE := 2:
3881                     END
```

```
3882                                ELSE
3883                                   EXIT := TRUE;
3884
3885                    END; (* CASE PATH *)
3886
3887            2 : IF GETENDREQUESTSYM
3888                    THEN
3889                       EXIT := TRUE
3890                    ELSE
3891                       SHOWERROR( ERRENDREQUEST );
3892
3893          END; (* CASE STATE *)
3894
3895 END; (* GETREQUEST *)
3896
3897 (*$Z *)
```

```
3898
3899 PROCEDURE TOKENIZE(VAR TEXT    : STRING:
3900                    VAR TEXTLEN : STRINGINDEX:
3901                    VAR NEWINFO : TOKENINFO:
3902                    VAR NEWLEN  : STRINGINDEX ):
3903
3904 (*************************************************************************)
3905 (*                                                                       *)
3906 (* TOKENIZE WILL CONVERT THE TEXT STRING INTO TOKEN FORM BY SEARCHING     *)
3907 (* FOR SYMBOLIC OCCURENCES OF A MARGIN, ELLIPSIS, OR JOKER. IF FOUND,    *)
3908 (* THE MARGIN SYMBOL IS CONVERTED DIRECTLY INTO THE INTERNAL REPRESEN-   *)
3909 (* TATION. THE JOKER AND ELLIPSIS SYMBOLS ARE CONVERTED INTO SINGLE      *)
3910 (* ELEMENT TOKENS.                                                       *)
3911 (*                                                                       *)
3912 (*************************************************************************)
3913
3914 VAR I : SYMINDEX:
3915     INDEX : STRINGINDEX:
3916
3917     HITMARGIN,
3918     HITJOKER,
3919     HITELLIPSIS : BOOLEAN:
3920
3921 BEGIN (* TOKENIZE *)
3922
3923    INDEX := NULLSTRLEN:
3924    NEWLEN := NULLSTRLEN:
3925
3926    WHILE (INDEX < TEXTLEN) AND (NEWLEN < MAXSTRLEN) AND NOT(ERROR) DO
3927       BEGIN
3928
3929          HITMARGIN := TRUE:
3930
3931          IF (INDEX + MARGINSYM.LEN > TEXTLEN) OR
3932             (MARGINSYM.LEN = NULLSYMLEN)
3933             THEN
3934                HITMARGIN := FALSE
3935             ELSE
3936                FOR I := 1 TO MARGINSYM.LEN DO
3937                   IF (TEXT [INDEX+I] <> MARGINSYM.VAL [I])
3938                      THEN
3939                         HITMARGIN := FALSE:
3940
3941          HITJOKER := TRUE:
3942
3943          IF (INDEX + JOKERSYM.LEN > TEXTLEN) OR
3944             (JOKERSYM.LEN = NULLSYMLEN)
3945             THEN
3946                HITJOKER := FALSE
3947             ELSE
3948                FOR I := 1 TO JOKERSYM.LEN DO
3949                   IF (TEXT [INDEX+I] <> JOKERSYM.VAL [I])
3950                      THEN
3951                         HITJOKER := FALSE:
3952
3953          HITELLIPSIS := TRUE:
3954
3955          IF (INDEX + ELLIPSISSYM.LEN > TEXTLEN) OR
3956             (ELLIPSISSYM.LEN = NULLSYMLEN)
3957             THEN
3958                HITELLIPSIS := FALSE
3959             ELSE
3960                FOR I := 1 TO ELLIPSISSYM.LEN DO
```

```
3961                IF (TEXT [INDEX+I] <> ELLIPSISSYM.VAL [I])
3962                   THEN
3963                      HITELLIPSIS := FALSE:
3964
3965
3966          IF (NEWLEN >= MAXSTRLEN - 4)
3967             THEN
3968                SHOWERROR( ERRTEXTLEN ):
3969
3970          IF HITMARGIN AND NOT(ERROR)
3971             THEN
3972                BEGIN
3973                   IF (INDEX + 1 = 1) AND CHECKLEFT
3974                      THEN
3975                         BEGIN
3976                            NEWINFO.SYM [NEWLEN+1] := CONSTANT:
3977                            NEWINFO.SYM [NEWLEN+2] := CONSTANT:
3978                            NEWINFO.CHR [NEWLEN+1] := LM [1]:
3979                            NEWINFO.CHR [NEWLEN+2] := LM [2]:
3980                            NEWLEN := NEWLEN + 2:
3981                         END
3982                      ELSE
3983                         IF (INDEX + MARGINSYM.LEN = TEXTLEN) AND
3984                            CHECKRIGHT
3985                            THEN
3986                               BEGIN
3987                                NEWINFO.SYM [NEWLEN+1] := CONSTANT:
3988                                NEWINFO.SYM [NEWLEN+2] := CONSTANT:
3989                                NEWINFO.CHR [NEWLEN+1] := RM [1]:
3990                                NEWINFO.CHR [NEWLEN+2] := RM [2]:
3991                                NEWLEN := NEWLEN + 2:
3992                               END
3993                            ELSE
3994                               BEGIN
3995                                NEWINFO.SYM [NEWLEN+1] := CONSTANT:
3996                                NEWINFO.SYM [NEWLEN+2] := CONSTANT:
3997                                NEWINFO.CHR [NEWLEN+1] := RM [1]:
3998                                NEWINFO.CHR [NEWLEN+2] := RM [2]:
3999                                NEWLEN := NEWLEN + 2:
4000                                NEWINFO.SYM [NEWLEN+1] := CONSTANT:
4001                                NEWINFO.SYM [NEWLEN+2] := CONSTANT:
4002                                NEWINFO.CHR [NEWLEN+1] := LM [1]:
4003                                NEWINFO.CHR [NEWLEN+2] := LM [2]:
4004                                NEWLEN := NEWLEN + 2:
4005                               END:
4006
4007                   INDEX := INDEX + MARGINSYM.LEN:
4008             END:
4009
4010          IF HITJOKER AND NOT(ERROR)
4011             THEN
4012                BEGIN
4013                   NEWLEN := NEWLEN + 1:
4014                   NEWINFO.SYM [NEWLEN] := WILDCARD:
4015                   INDEX := INDEX + JOKERSYM.LEN:
4016
4017                   IF (ALPHABET = ASCII)
4018                      THEN
4019                         SHOWERROR( ERRASCIIJOKER ):
4020             END:
4021
4022          IF HITELLIPSIS AND NOT(ERROR)
4023             THEN
4024                BEGIN
```

```
4025                    NEWLEN := NEWLEN + 1;
4026                    NEWINFO.SYM [NEWLEN] := FREESTRING;
4027                    INDEX := INDEX + ELLIPSISSYM.LEN;
4028                END;
4029
4030          IF NOT(HITMARGIN OR HITJOKER OR HITELLIPSIS) AND
4031             NOT(ERROR)
4032             THEN
4033                BEGIN
4034                   NEWLEN := NEWLEN + 1;
4035                   INDEX := INDEX + 1;
4036                   NEWINFO.SYM [NEWLEN] := CONSTANT;
4037                   NEWINFO.CHR [NEWLEN] := TEXT [INDEX];
4038
4039                   IF (INDEX < TEXTLEN)
4040                      THEN
4041                         IF ((TEXT [INDEX] = LM [1]) AND
4042                             (TEXT [INDEX+1] = LM [2])) OR
4043                             ((TEXT [INDEX] = RM[1]) AND
4044                             (TEXT [INDEX+1] = RM [2]))
4045                             THEN
4046                               SHOWERROR( ERRCHARACTERS );
4047                END;
4048
4049       END; (* WHILE *)
4050
4051    IF (NEWLEN <> NULLSYMLEN) AND NOT(ERROR)
4052       THEN
4053          IF (NEWINFO.SYM [1] = FREESTRING) OR
4054             (NEWINFO.SYM [NEWLEN] = FREESTRING)
4055             THEN
4056                SHOWERROR( ERRELLPOSITION );
4057
4058 END; (* TOKENIZE *)
4059
4060 (*$Z *)
```

```
4061
4062 PROCEDURE PARSEREQUEST;
4063
4064 (*************************************************************************)
4065 (*                                                                       *)
4066 (* PARSEREQUEST WILL READ AN INPUT LINE AND ATTEMPT TO PARSE A LEGAL     *)
4067 (* REQUEST. IF A SYNTACTICALLY CORRECT REQUEST IS FOUND, SEMANTIC        *)
4068 (* CHECKS CONCERNING THE MARGIN, ELLIPSIS AND JOKER SYMBOLS ARE          *)
4069 (* PERFORMED BEFORE THE REQUEST IS ACCEPTED.                             *)
4070 (*                                                                       *)
4071 (*************************************************************************)
4072
4073 VAR I : INTEGER;
4074
4075     NUMPATJOK,
4076     NUMREPJOK,
4077
4078     NUMPATELL,
4079     NUMREPELL : STRINGINDEX;
4080
4081 BEGIN (* PARSEREQUEST *)
4082
4083    REPEAT
4084
4085       REQUEST := NULLREQUEST;
4086
4087       PATLEN  := NULLSTRLEN;
4088       REPLEN  := NULLSTRLEN;
4089
4090       PATTERN     := NULLPATTERN;
4091       REPLACEMENT := NULLPATTERN;
4092
4093       IF (ERROR OR ENDOFLINE)
4094          THEN
4095             REPEAT
4096
4097                GETLINE( REQPROMPT, TEXTLINE, LINELENGTH );
4098                CURLINEPOS := 1;
4099                NEXTCHAR := TEXTLINE [1];
4100
4101                ENDOFLINE := FALSE;
4102                SHOWINPUT := FALSE;
4103                SHOWLINE  := FALSE;
4104                ERROR     := FALSE;
4105
4106             UNTIL (LINELENGTH <> NULLSTRLEN) AND NOT(INTERRUPT);
4107
4108       IF NOT(GETREQUEST)
4109          THEN
4110             SHOWERROR( ERRNOREQUEST );
4111
4112       WITH REQUEST, PATTERN DO
4113          IF (NAME IN [FORWARD..INSERT]) AND NOT(ERROR)
4114             THEN
4115                BEGIN
4116                   CHECKLEFT  := TRUE;
4117                   CHECKRIGHT := TRUE;
4118
4119                   TOKENIZE (TEXTSTR, TEXTLEN, PATTERN, PATLEN);
4120
4121                   IF (TEXTLEN = NULLSTRLEN)
4122                      THEN
4123                         IF (NAME = MAKE)
```

```
4124                            THEN
4125                               SHOWERROR( ERRMISSINGTEXT )
4126                            ELSE
4127                               IF (REPEATER = REPEATALL)
4128                                THEN
4129                                 BEGIN
4130                                  SYM [1] := FREESTRING:
4131                                  PATLEN := 1:
4132                                 END
4133                                ELSE
4134                                 BEGIN
4135                                  SYM [1] := CONSTANT:
4136                                  SYM [2] := CONSTANT:
4137                                  SYM [3] := FREESTRING:
4138                                  SYM [4] := CONSTANT:
4139                                  SYM [5] := CONSTANT:
4140                                  CHR [1] := LM [1]:
4141                                  CHR [2] := LM [2]:
4142                                  CHR [4] := RM [1]:
4143                                  CHR [5] := RM [2]:
4144                                  PATLEN := 5:
4145                                 END:
4146
4147                     CHECKLEFT  := FALSE:
4148                     CHECKRIGHT := FALSE:
4149
4150                     IF (PATLEN >= 2)
4151                        THEN
4152                           BEGIN
4153                              IF (CHR [1] = LM [1]) AND
4154                                 (CHR [2] = LM [2])
4155                                 THEN
4156                                  CHECKLEFT := TRUE:
4157
4158                              IF (CHR [PATLEN-1] = RM [1]) AND
4159                                 (CHR [PATLEN]   = RM [2])
4160                                 THEN
4161                                  CHECKRIGHT  := TRUE:
4162                           END:
4163
4164                     NUMPATJOK := 0:
4165                     NUMPATELL := 0:
4166
4167                     IF (PATLEN <> NULLSTRLEN)
4168                        THEN
4169                           FOR I := 1 TO PATLEN DO
4170                              IF (SYM [I] = WILDCARD)
4171                                 THEN
4172                                  NUMPATJOK := NUMPATJOK + 1
4173                                 ELSE
4174                                  IF (SYM [I] = FREESTRING)
4175                                   THEN
4176                                    NUMPATELL := NUMPATELL + 1:
4177               END:
4178
4179         WITH REQUEST, REPLACEMENT DO
4180            IF (NAME = MAKE) AND NOT(ERROR)
4181               THEN
4182                  BEGIN
4183                     TOKENIZE (NEWTEXTSTR, NEWTEXTLEN,
4184                               REPLACEMENT, REPLEN):
4185
4186                     IF CHECKLEFT AND NOT(ERROR)
4187                        THEN
```

```
4188                        IF (REPLEN < 2)
4189                           THEN
4190                              SHOWERROR( ERRMARGINSYM )
4191                           ELSE
4192                              IF (CHR [1] <> LM [1]) OR
4193                                 (CHR [2] <> LM [2])
4194                                 THEN
4195                                  SHOWERROR( ERRMARGINSYM );
4196
4197                     IF CHECKRIGHT AND NOT(ERROR)
4198                        THEN
4199                           IF (REPLEN < 2)
4200                              THEN
4201                                 SHOWERROR( ERRMARGINSYM )
4202                              ELSE
4203                                 IF (CHR [REPLEN-1] <> RM [1]) OR
4204                                    (CHR [REPLEN]   <> RM [2])
4205                                    THEN
4206                                     SHOWERROR( ERRMARGINSYM );
4207
4208                     IF (REPLEN >= 2) AND NOT(ERROR)
4209                        THEN
4210                           IF (CHR [1] = LM [1]) AND
4211                              (CHR [2] = LM [2]) AND
4212                              NOT(CHECKLEFT)
4213                              THEN
4214                                 SHOWERROR( ERRMARGINSYM );
4215
4216                     IF (REPLEN >= 2) AND NOT(ERROR)
4217                        THEN
4218                           IF (CHR [REPLEN-1] = RM [1]) AND
4219                              (CHR [REPLEN]   = RM [2]) AND
4220                              NOT(CHECKRIGHT)
4221                              THEN
4222                                 SHOWERROR( ERRMARGINSYM );
4223
4224                     NUMREPJOK := 0;
4225                     NUMREPELL := 0;
4226
4227                     IF (REPLEN <> NULLSTRLEN)
4228                        THEN
4229                           FOR I := 1 TO REPLEN DO
4230                              IF (SYM [I] = WILDCARD)
4231                                 THEN
4232                                  NUMREPJOK := NUMREPJOK + 1
4233                                 ELSE
4234                                  IF (SYM [I] = FREESTRING)
4235                                   THEN
4236                                    NUMREPELL := NUMREPELL + 1;
4237
4238                     IF (NUMREPJOK > NUMPATJOK) AND NOT(ERROR)
4239                        THEN
4240                           SHOWERROR( ERRJOKERSYM );
4241
4242                     IF (NUMREPELL > NUMPATELL) AND NOT(ERROR)
4243                        THEN
4244                           SHOWERROR( ERRELLIPSISSYM );
4245                  END;
4246
4247     UNTIL NOT(ERROR) AND NOT(INTERRUPT);
4248
4249 END; (* PARSEREQUEST *)
4250
4251 (*$Z *)
```

```
4252
4253 PROCEDURE SECURITYCHECK(    CHK : CHECKPROBLEM:
4254                         VAR ANSWER : KEYSYMBOL ):
4255
4256 (*************************************************************************)
4257 (*                                                                       *)
4258 (* SECURITYCHECK WILL DISPLAY A MESSAGE CONCERNING THE PROBLEM.          *)
4259 (* DEPENDING ON THE RESPONSE, THE ACTIONS OF THE REQUEST WILL EITHER     *)
4260 (* BE PERFORMED OR UNDONE.                                               *)
4261 (*                                                                       *)
4262 (*************************************************************************)
4263
4264 VAR OK : BOOLEAN:
4265     I : INTEGER:
4266
4267 BEGIN (* SECURITYCHECK *)
4268
4269    IF (CHK IN [CHKSINGLECHAR, CHKDOUBLECHAR])
4270       THEN
4271          SHOWMESSAGE( RESTRICTION ):
4272
4273    IF (CHK <> DOREQUEST)
4274       THEN
4275          WRITE (MSGPREFIX):
4276
4277    CASE CHK OF
4278
4279       ELIMINATEFILE,
4280       OVERWRITEFILE : BEGIN
4281                          IF (CHK = OVERWRITEFILE)
4282                             THEN
4283                                WRITE ('EXISTING '):
4284                          PRINT (FILESYM):
4285                          WRITE (' '):
4286                          I := 1:
4287                          WHILE (REQUEST.FILENAME[I] <> BLANK) DO
4288                             BEGIN
4289                                WRITE (REQUEST.FILENAME[I]):
4290                                I := I + 1:
4291                             END:
4292                          WRITE (' WILL BE LOST AFTER '):
4293                          PRINT (REQUEST.NAME):
4294                          WRITE (' REQUEST '):
4295                       END:
4296
4297       FORGETFILE,
4298       OKTOQUIT      : BEGIN
4299                          WRITE ('CHANGES TO CURRENT '):
4300                          PRINT (FILESYM):
4301                          WRITE (' '):
4302                          I := 1:
4303                          WHILE (CURFILENAME[I] <> BLANK) DO
4304                             BEGIN
4305                                WRITE (CURFILENAME[I]):
4306                                I := I + 1:
4307                             END:
4308                          WRITE (' WILL BE IGNORED AFTER '):
4309                          PRINT (REQUEST.NAME):
4310                          WRITE (' REQUEST'):
4311                       END:
4312
4313       CHKSINGLECHAR : BEGIN
4314                          WRITE ('THE CHARACTER '''):
```

```
4315                              IF (CURALPHABET <> ASCII)
4316                                 THEN
4317                                    WRITE (OUTMARKER)
4318                                 ELSE
4319                                    WRITE (ASCOUTMARKER):
4320                              WRITE ('" CANNOT BE THE '):
4321                              WRITE ('FIRST OR LAST CHARACTER '):
4322                              WRITE ('ON A LINE '):
4323                           END:
4324
4325        CHKDOUBLECHAR   : BEGIN
4326                              WRITE ('THE CHARACTERS "'):
4327                              WRITE (OUTMARKER):
4328                              WRITE ('" AND "'):
4329                              IF (CURALPHABET <> ASCII)
4330                                 THEN
4331                                    WRITE (INMARKER)
4332                                 ELSE
4333                                    WRITE (ASCINMARKER):
4334                              WRITE ('" CANNOT '):
4335                              WRITE ('OCCUR IN SEQUENCE '):
4336                              WRITE ('ON A LINE '):
4337                           END:
4338
4339        DOREQUEST       : ;  (* MESSAGE PRINTED ALREADY *)
4340
4341     END: (* CASE *)
4342
4343     IF (CHK <> DOREQUEST)
4344        THEN
4345           WRITELN:
4346
4347
4348     WRITE (MSGPREFIX):
4349
4350     CASE CHK OF
4351
4352        CHKSINGLECOLON,
4353        CHKDOUBLECOLON : BEGIN
4354                              WRITE ('CAN A BLANK BE INSERTED '):
4355                              WRITE ('TO CORRECT THIS PROBLEM? '):
4356                           END:
4357
4358        OVERWRITEFILE,
4359        ELIMINATEFILE,
4360        FORGETFILE,
4361        OKTOQUIT,
4362        DOREQUEST       : BEGIN
4363                              WRITE ('SHOULD THE REQUEST BE '):
4364                              WRITE ('PERFORMED ANYWAY? '):
4365                           END:
4366
4367     END: (* CASE *)
4368
4369     WRITELN:
4370
4371     GETRESPONSE (ANSWER):
4372
4373     IF (ANSWER = YES)
4374        THEN
4375           ERROR := FALSE
4376        ELSE
4377           ERROR := TRUE:
4378
```

```
4379     SHOWINPUT := TRUE;
4380
4381 END; (* SECURITYCHECK *)
4382
4383 (*$Z *)


4384
4385 PROCEDURE PUTINTO( FILENAME : ALFA );
4386
4387 (*************************************************************************)
4388 (*                                                                       *)
4389 (* PUTINTO WILL RETURN THE SPECIFIED FILE TO THE USER'S CATALOG. IF      *)
4390 (* THE FILE CANNOT BE SAVED, A MESSAGE WILL BE DISPLAYED.                *)
4391 (*                                                                       *)
4392 (*************************************************************************)
4393
4394 CONST EMPTYLINE = '          ';
4395
4396 VAR OK : BOOLEAN;
4397
4398 BEGIN (* PUTINTO *)
4399
4400     IF NOT(VALIDINFO)
4401         THEN
4402             BEGIN
4403               REWRITE (SCRATCH);
4404               VALIDINFO := TRUE;
4405               WRITELN (SCRATCH, EMPTYLINE);
4406             END;
4407
4408     OK := FALSE;
4409
4410     SAVE (FILENAME, SCRATCH, OK);
4411
4412     IF NOT(OK)
4413         THEN
4414             SHOWERROR( ERRCANTSAVE );
4415
4416 END; (* PUTINTO *)
4417
4418 (*$Z *)
```

```
4419
4420 PROCEDURE SETUP( FILENAME : ALFA ):
4421
4422 (*************************************************************************)
4423 (*                                                                       *)
4424 (* SETUP WILL RETRIEVE THE SPECIFIED FILE FROM THE USER'S CATALOG.       *)
4425 (* IF THE FILE IS NOT FOUND, AN EMPTY FILE WILL BE SET UP AND A          *)
4426 (* MESSAGE WILL BE DISPLAYED.                                            *)
4427 (*                                                                       *)
4428 (*************************************************************************)
4429
4430 VAR OK : BOOLEAN:
4431
4432 BEGIN (* SETUP *)
4433
4434    OK := FALSE:
4435
4436    FIND (FILENAME, SCRATCH, OK):
4437
4438    IF OK
4439       THEN
4440          BEGIN
4441             FILESTATUS := OLD:
4442             VALIDINFO := TRUE:
4443          END
4444       ELSE
4445          BEGIN
4446             SHOWMESSAGE( NOFILEFOUND ):
4447             REWRITE (SCRATCH):
4448             VALIDINFO := FALSE:
4449
4450             IF (REQUEST.NAME <> ASSUME)
4451                THEN
4452                   SHOWMESSAGE( NEWFILE ):
4453             FILESTATUS := NEW:
4454          END:
4455
4456 END: (* SETUP *)
4457
4458 (*$7 *)
```

```
4459
4460 PROCEDURE READFILE;
4461
4462 (*************************************************************************)
4463 (*                                                                       *)
4464 (* READFILE WILL READ THE INFORMATION STORED IN THE SCRATCH FILE         *)
4465 (* INTO THE INCORETEXT. IN THE PROCESS, LEFT AND RIGHT INTERNAL          *)
4466 (* MARGIN MARKERS WILL BE INSERTED.                                      *)
4467 (*                                                                       *)
4468 (*************************************************************************)
4469
4470 TYPE CHECKTYPE = (SINGLE, DOUBLE);
4471
4472 VAR INDEX : INCOREINDEX;
4473
4474     CHR   : CHAR;
4475
4476     FIRSTINDEX,
4477     LASTLINE : INCOREINDEX;
4478
4479     CHECK : ARRAY [ CHECKTYPE ] OF BOOLEAN;
4480
4481     FAIL,
4482     FOUND : BOOLEAN;
4483
4484 BEGIN (* READFILE *)
4485
4486    IF VALIDINFO
4487       THEN
4488          RESET (SCRATCH);
4489
4490    FAIL := FALSE;
4491
4492    LINESFOUND := 0;
4493
4494    CHECK [ SINGLE ] := FALSE;
4495    CHECK [ DOUBLE ] := FALSE;
4496
4497    IF (HOLE.FIRST = NULLPOINTER)
4498       THEN
4499          FAIL := TRUE
4500       ELSE
4501          INDEX := HOLE.FIRST - 1;
4502
4503    LASTLINE := NULLPOINTER;
4504
4505    WHILE NOT EOF(SCRATCH) AND NOT(FAIL) DO
4506       BEGIN
4507
4508          IF (INDEX > HOLE.LAST - 2)
4509             THEN
4510                FAIL := TRUE
4511             ELSE
4512                BEGIN
4513                   INCORETEXT [INDEX + 1] := LM [1];
4514                   INCORETEXT [INDEX + 2] := LM [2];
4515                   INDEX := INDEX + 2;
4516                   FIRSTINDEX := INDEX;
4517                END;
4518
4519          IF (SCRATCH^ = LM [1])
4520             THEN
4521                IF (INDEX > HOLE.LAST - 1)
```

```
4522                THEN
4523                   FAIL := TRUE
4524                ELSE
4525                   BEGIN
4526                      CHECK [ SINGLE ] := TRUE;
4527                      INCORETEXT [INDEX + 1] := BLANK;
4528                      INDEX := INDEX + 1;
4529                   END;
4530
4531          WHILE NOT EOLN(SCRATCH) AND (INDEX <= HOLE.LAST-1) DO
4532             BEGIN
4533
4534                READ (SCRATCH, CHR);
4535                INCORETEXT [INDEX + 1] := CHR;
4536                INDEX := INDEX + 1;
4537
4538                IF (CHR = LM [1]) AND (SCRATCH^ = LM [2]) OR
4539                   (CHR = RM [1]) AND (SCRATCH^ = RM [2])
4540                   THEN
4541                      IF (INDEX > HOLE.LAST - 1)
4542                         THEN
4543                            FAIL := TRUE
4544                         ELSE
4545                            BEGIN
4546                               CHECK [ DOUBLE ] := TRUE;
4547                               INCORETEXT [INDEX + 1] := BLANK;
4548                               INDEX := INDEX + 1;
4549                            END;
4550
4551             END; (* WHILE NOT EOLN *)
4552
4553          READLN (SCRATCH);
4554
4555          FOUND := FALSE;
4556
4557          WHILE (INDEX <> FIRSTINDEX) AND NOT FOUND DO
4558             IF (INCORETEXT [INDEX] <> BLANK)
4559                THEN
4560                   FOUND := TRUE
4561                ELSE
4562                   INDEX := INDEX - 1;
4563
4564          IF (CHR = RM [2])
4565             THEN
4566                IF (INDEX > HOLE.LAST)
4567                   THEN
4568                      FAIL := TRUE
4569                   ELSE
4570                      BEGIN
4571                         CHECK [ SINGLE ] := TRUE;
4572                         INCORETEXT [INDEX + 1] := BLANK;
4573                         INDEX := INDEX + 1;
4574                      END;
4575
4576          IF (INDEX > HOLE.LAST - 2)
4577             THEN
4578                FAIL := TRUE
4579             ELSE
4580                BEGIN
4581                   INCORETEXT [INDEX + 1] := RM [1];
4582                   INCORETEXT [INDEX + 2] := RM [2];
4583                   INDEX := INDEX + 2;
4584                   LASTLINE := INDEX;
4585                   LINESFOUND := LINESFOUND + 1;
```

```
4586                    END;
4587
4588            END; (* WHILE NOT EOF *)
4589
4590
4591
4592
4593        ANSWER := YES;
4594
4595        IF NOT EOF(SCRATCH)
4596            THEN
4597                BEGIN
4598
4599                    SHOWERROR( ERRNOTALLLINES );
4600
4601                    IF (LINESFOUND > 0)
4602                        THEN
4603                            BEGIN
4604                                SECURITYCHECK( DOFEQUEST, ANSWER );
4605                                FILESTATUS := NEW;
4606                            END
4607                        ELSE
4608                            ANSWER := NO;
4609
4610                END;
4611
4612        IF CHECK [ SINGLE ] AND
4613            (ANSWER = YES)
4614            THEN
4615                SECURITYCHECK( CHKSINGLECHAR, ANSWER );
4616
4617        IF CHECK [ DOUBLE ] AND
4618            (ANSWER = YES)
4619            THEN
4620                SECURITYCHECK( CHKDOUBLECHAR, ANSWER );
4621
4622        IF (LASTLINE <> NULLPOINTER) AND
4623            (ANSWER = YES)
4624            THEN
4625                BEGIN
4626
4627                    IF (UPPERSEQ.FIRST = NULLPOINTER)
4628                        THEN
4629                            UPPERSEQ.FIRST := HOLE.FIRST;
4630
4631                    UPPERSEQ.LAST := LASTLINE;
4632
4633                    IF (LASTLINE = MAXNUMCHARS)
4634                        THEN
4635                            HOLE := NULLSEQ
4636                        ELSE
4637                            HOLE.FIRST := LASTLINE + 1;
4638
4639                END;
4640
4641    END; (* READFILE *)
4642
4643    (*$7 *)
```

```
4644
4645 PROCEDURE WRITEFILE( VAR LINES : INCOREPTRS ):
4646
4647 (*****************************************************************)
4648 (*                                                               *)
4649 (* WRITEFILE WILL WRITE THE INCORETEXT OUT TO THE SCRATCH FILE. IN *)
4650 (* THE PROCESS, THE INTERNAL MARGIN MARKERS ARE REMOVED.         *)
4651 (*                                                               *)
4652 (*****************************************************************)
4653
4654 VAR INDEX : INCOREINDEX:
4655
4656 BEGIN (* WRITEFILE *)
4657
4658    IF (LINES.FIRST <> NULLPOINTER)
4659       THEN
4660          BEGIN
4661
4662             VALIDINFO := TRUE:
4663             INDEX := LINES.FIRST - 1:
4664
4665             WHILE (INDEX <> LINES.LAST) DO
4666                BEGIN
4667
4668                   INDEX := INDEX + 2:
4669
4670                   WHILE (INCORETEXT [INDEX + 1] <> RM [1]) OR
4671                         (INCORETEXT [INDEX + 2] <> RM [2]) DO
4672                      BEGIN
4673                         INDEX := INDEX + 1:
4674                         WRITE (SCRATCH, INCORETEXT [INDEX]):
4675                      END:
4676
4677                   INDEX := INDEX + 2;
4678
4679                   WRITELN (SCRATCH):
4680                   LINESFOUND := LINESFOUND + 1:
4681
4682                END: (* WHILE *)
4683
4684          END;
4685
4686 END: (* WRITEFILE *)
4687
4688 (*$7 *)
```

```
4689
4690 PROCEDURE CORRSTR(     LINES : INCORETRS:
4691                    VAR HEAD, HIT, TAIL : INCORETRS:
4692                    VAR FIRSTCHAR, LASTCHAR : STRINGINDEX ):
4693
4694 (*****************************************************************)
4695 (*                                                               *)
4696 (* CORRSTR WILL ATTEMPT TO MATCH THE SPECIFIED STRING IN THE     *)
4697 (* INCORETEXT. IF FOUND, THE HEAD, HIT, AND TAIL POINTERS ARE SET*)
4698 (* APPROPRIATELY. IN THE PROCESS, MATCHED JOKER CHARACTERS ARE   *)
4699 (* SAVED. IF NOT FOUND, HIT IS SET TO NULL.                      *)
4700 (*                                                               *)
4701 (*****************************************************************)
4702
4703 VAR INCREMENT : -1..1:
4704
4705     TEXTINDEX : STRINGINDEX:
4706
4707     INDEX,
4708     FIRSTHIT,
4709     FIRSTINDEX,
4710     LASTINDEX   : INCOREINDEX:
4711
4712     MISMATCH,
4713     LASTPOSSIBLE,
4714     FOUND : BOOLEAN:
4715
4716 BEGIN (* CORRSTR *)
4717
4718    IF (LINES.FIRST <= LINES.LAST)
4719       THEN
4720          INCREMENT := 1
4721       ELSE
4722          INCREMENT := -1:
4723
4724    FIRSTINDEX := LINES.FIRST:
4725    LASTINDEX  := LINES.LAST:
4726
4727    FOUND := FALSE:
4728    INDEX := FIRSTINDEX:
4729
4730    LASTPOSSIBLE := FALSE:
4731
4732    IF (FIRSTINDEX <> NULLPOINTER)
4733       THEN
4734          REPEAT
4735
4736             MISMATCH := FALSE:
4737             TEXTINDEX := FIRSTCHAR:
4738             FIRSTHIT := INDEX:
4739
4740             WITH PATTERN DO
4741                WHILE (((SYM [TEXTINDEX] = CONSTANT) AND
4742                      (CHR [TEXTINDEX] = INCORETEXT [INDEX])) OR
4743
4744                      (SYM [TEXTINDEX] = WILDCARD)) AND
4745
4746                      NOT(FOUND OR LASTPOSSIBLE OR MISMATCH) DO
4747                   BEGIN
4748
4749                      IF (SYM [TEXTINDEX] = WILDCARD)
4750                         THEN
4751                            IF (INCORETEXT [INDEX] = OUTMARKER) OR
```

```
4752                                (INCORETEXT [INDEX] = INMARKER)
4753                                THEN
4754                                 MISMATCH := TRUE
4755                                ELSE
4756                                 JOK[TEXTINDEX] := INCORETEXT[INDEX]:
4757
4758                          IF (TEXTINDEX = LASTCHAR) AND NOT(MISMATCH)
4759                             THEN
4760                                FOUND := TRUE
4761                             ELSE
4762                                IF (INDEX = LASTINDEX)
4763                                THEN
4764                                 LASTPOSSIBLE := TRUE
4765                                ELSE
4766                                 BEGIN
4767                                  TEXTINDEX := TEXTINDEX + INCREMENT:
4768                                  INDEX := INDEX + INCREMENT:
4769                                 END:
4770
4771                       END: (* WHILE *)
4772
4773                IF (INDEX = LASTINDEX)
4774                   THEN
4775                      LASTPOSSIBLE := TRUE
4776                   ELSE
4777                      IF NOT(FOUND) OR MISMATCH
4778                         THEN
4779                            INDEX := FIRSTHIT + INCREMENT:
4780
4781             UNTIL (FOUND OR LASTPOSSIBLE):
4782
4783
4784        IF NOT(FOUND)
4785           THEN
4786              BEGIN
4787                 HEAD := NULLSEQ:
4788                 HIT  := NULLSEQ:
4789                 TAIL := NULLSEQ:
4790              END
4791           ELSE
4792              BEGIN
4793                 IF (FIRSTHIT = FIRSTINDEX)
4794                 THEN
4795                    HEAD := NULLSEQ
4796                 ELSE
4797                    BEGIN
4798                       HEAD.FIRST := FIRSTINDEX:
4799                       HEAD.LAST  := FIRSTHIT - INCREMENT:
4800                    END:
4801
4802                 HIT.FIRST := FIRSTHIT:
4803                 HIT.LAST  := INDEX:
4804
4805                 IF (INDEX = LASTINDEX)
4806                 THEN
4807                    TAIL := NULLSEQ
4808                 ELSE
4809                    BEGIN
4810                       TAIL.FIRST := INDEX + INCREMENT:
4811                       TAIL.LAST  := LASTINDEX:
4812                    END:
4813              END:
4814
4815  END: (* CORRSTR *)
```

```
4816
4817 (*7 *)
4818
4819 PROCEDURE CORRFILEFRAG(    LINES : INCOREPTRS:
4820                        VAR HEAD, HIT, TAIL : INCOREPTRS ):
4821
4822 (*****************************************************************)
4823 (*                                                               *)
4824 (* CORRFILEFRAG ATTEMPTS TO MATCH THE SPECIFIED LINES IN THE     *)
4825 (* INCORETEXT. IN THE PROCESS, ANY ELLIPSIS CHARACTERS ARE SET.  *)
4826 (* IF FOUND, THE HEAD, HIT, AND TAIL POINTERS ARE SET APPROPRIATELY. *)
4827 (* IF NOT FOUND, HIT IS SET TO NULL.                             *)
4828 (*                                                               *)
4829 (*****************************************************************)
4830
4831 VAR INCREMENT : -1..1:
4832
4833     INDEX : INTEGER:
4834
4835     FIRSTINDEX,
4836     LASTINDEX,
4837     FIRSTCHAR,
4838     LASTCHAR : STRINGINDEX:
4839
4840     FIRSTLINE,
4841     LASTLINE : INCOREINDEX:
4842
4843     HITSEQ : INCOREPTRS:
4844
4845     FOUND : BOOLEAN:
4846
4847 BEGIN (* CORRFILEFRAG *)
4848
4849    IF (LINES.FIRST <= LINES.LAST)
4850       THEN
4851          BEGIN
4852             INCREMENT  := 1:
4853             FIRSTINDEX := 1:
4854             LASTINDEX  := PATLEN:
4855          END
4856       ELSE
4857          BEGIN
4858             INCREMENT  := -1:
4859             FIRSTINDEX := PATLEN:
4860             LASTINDEX  := 1:
4861          END:
4862
4863    FIRSTLINE := LINES.FIRST:
4864    LASTLINE  := LINES.LAST:
4865
4866    HITSEQ := NULLSEQ:
4867    HEAD   := NULLSEQ:
4868    HIT    := NULLSEQ:
4869    TAIL   := NULLSEQ:
4870
4871    INDEX := FIRSTINDEX - INCREMENT:
4872
4873    WITH PATTERN DO
4874       REPEAT
4875
4876          FIRSTCHAR := INDEX + INCREMENT:
4877          INDEX := FIRSTCHAR:
4878          FOUND := FALSE:
4879
```

```
4880          WHILE NOT(FOUND) DO
4881             IF (INDEX = LASTINDEX) OR
4882                (SYM [INDEX] = FREESTRING)
4883                THEN
4884                   FOUND := TRUE
4885                ELSE
4886                   INDEX := INDEX + INCREMENT;
4887
4888          IF (INDEX = FIRSTCHAR) OR
4889             (INDEX = LASTINDEX)
4890             THEN
4891                LASTCHAR := INDEX
4892             ELSE
4893                LASTCHAR := INDEX - INCREMENT;
4894
4895          IF (SYM [FIRSTCHAR] <> FREESTRING)
4896             THEN
4897                CORRSTR( LINES, HEAD, HIT, TAIL, FIRSTCHAR, LASTCHAR );
4898
4899          IF (FIRSTCHAR <> FIRSTINDEX)
4900             THEN
4901                IF (SYM [FIRSTCHAR-1] = FREESTRING)
4902                   THEN
4903                      ELL [FIRSTCHAR-1] := HEAD;
4904
4905          IF (HITSEQ.FIRST = NULLPOINTER)
4906             THEN
4907                HITSEQ := HIT
4908             ELSE
4909                IF (HIT.FIRST = NULLPOINTER)
4910                   THEN
4911                      HITSEQ := NULLSEQ
4912                   ELSE
4913                      HITSEQ.LAST := HIT.LAST;
4914
4915          LINES := TAIL;
4916
4917       UNTIL (HIT.FIRST = NULLPOINTER) OR (LASTCHAR = LASTINDEX);
4918
4919
4920    WITH PATTERN DO
4921       IF (SYM [FIRSTINDEX] = FREESTRING)
4922          THEN
4923             BEGIN
4924                HITSEQ.FIRST := FIRSTLINE;
4925                HITSEQ.LAST  := LASTLINE;
4926             END;
4927
4928    IF (HITSEQ.FIRST = NULLPOINTER)
4929       THEN
4930          BEGIN
4931             HEAD := NULLSEQ;
4932             HIT  := NULLSEQ;
4933             TAIL := NULLSEQ;
4934          END
4935       ELSE
4936          BEGIN
4937
4938             IF (HITSEQ.FIRST = FIRSTLINE)
4939                THEN
4940                   HEAD := NULLSEQ
4941                ELSE
4942                   BEGIN
4943                      HEAD.FIRST := FIRSTLINE;
```

```
4944                        HEAD.LAST  := HITSEQ.FIRST - INCREMENT;
4945                      END;
4946
4947              HIT := HITSEQ;
4948
4949              TAIL := LINES;
4950
4951            END;
4952
4953  END; (* CORRFILEFRAG *)


4954
4955  (*$Z *)
4956
4957  PROCEDURE REVERSE( VAR LINES : INCOREPTRS );
4958
4959  (*****************************************************************)
4960  (*                                                               *)
4961  (* REVERSE WILL SWAP THE SET OF POINTERS SO THAT THE FIRST POINTS *)
4962  (* TO WHAT WAS THE LAST AND VICE VERSE.                          *)
4963  (*                                                               *)
4964  (*****************************************************************)
4965
4966  VAR TEMP : INCOREPTRS;
4967
4968  BEGIN (* REVERSE *)
4969
4970     TEMP.FIRST := LINES.LAST;
4971     TEMP.LAST  := LINES.FIRST;
4972
4973     LINES := TEMP;
4974
4975  END; (* REVERSE *)
4976
4977  (*$Z *)
```

```
4978
4979 PROCEDURE APPEND( VAR NEWLINES, OLDLINES : INCOREPTRS ):
4980
4981 (*************************************************************)
4982 (*                                                           *)
4983 (* APPEND WILL ADD THE SPECIFIED NEW LINES TO THE OLD LINES. IN *)
4984 (* ADDITION, THE NEWLINE POINTER IS SET TO NULL.             *)
4985 (*                                                           *)
4986 (*************************************************************)
4987
4988 VAR TOPLINES,
4989     BOTTOMLINES : INCOREPTRS:
4990
4991 BEGIN (* APPEND *)
4992
4993    IF (NEWLINES.FIRST <= OLDLINES.FIRST)
4994       THEN
4995          BEGIN
4996             TOPLINES := NEWLINES:
4997             BOTTOMLINES := OLDLINES:
4998          END
4999       ELSE
5000          BEGIN
5001             TOPLINES := OLDLINES:
5002             BOTTOMLINES := NEWLINES:
5003          END:
5004
5005    IF (TOPLINES.FIRST <> NULLPOINTER)
5006       THEN
5007          IF (BOTTOMLINES.FIRST = NULLPOINTER)
5008          THEN
5009             BOTTOMLINES := TOPLINES
5010          ELSE
5011             BOTTOMLINES.FIRST := TOPLINES.FIRST:
5012
5013    OLDLINES := BOTTOMLINES:
5014    NEWLINES := NULLSEQ:
5015
5016 END: (* APPEND *)
5017
5018 (*$Z *)
```

```
5019
5020 PROCEDURE GETFIRST(     LINES : INCOREPTRS:
5021                       VAR FIRSTLINE, REMAININGLINES : INCOREPTRS ):
5022
5023 (**************************************************************)
5024 (*                                                            *)
5025 (* GETFIRST WILL RETURN THE FIRST WHOLE LINE IN THE SPECIFIED FILE.   *)
5026 (*                                                            *)
5027 (**************************************************************)
5028
5029 VAR INDEX : INCOREINDEX:
5030
5031 BEGIN (* GETFIRST *)
5032
5033    IF (LINES.FIRST <> NULLPOINTER)
5034       THEN
5035          BEGIN
5036             INDEX := LINES.FIRST + 1:
5037
5038             WHILE (INCORETEXT [INDEX + 1] <> RM [1]) OR
5039                   (INCORETEXT [INDEX + 2] <> RM [2]) DO
5040                INDEX := INDEX + 1:
5041
5042             INDEX := INDEX + 2:
5043          END:
5044
5045    IF (LINES.FIRST = NULLPOINTER)
5046       THEN
5047          BEGIN
5048             FIRSTLINE := NULLSEQ:
5049             REMAININGLINES := NULLSEQ:
5050          END
5051       ELSE
5052          BEGIN
5053
5054             FIRSTLINE.FIRST := LINES.FIRST:
5055             FIRSTLINE.LAST  := INDEX:
5056
5057             IF (INDEX = LINES.LAST)
5058                THEN
5059                   REMAININGLINES := NULLSEQ
5060                ELSE
5061                   BEGIN
5062                      REMAININGLINES.FIRST := INDEX + 1:
5063                      REMAININGLINES.LAST  := LINES.LAST:
5064                   END:
5065          END:
5066
5067 END: (* GETFIRST *)
5068
5069 (*97 *)
```

```
5070
5071 PROCEDURE GETLAST(    LINES : INCOREPTRS:
5072                   VAR LEADINGLINES, LASTLINE : INCOREPTRS ):
5073
5074 (*****************************************************************)
5075 (*                                                               *)
5076 (* GETLAST WILL RETURN THE LAST WHOLE LINE IN THE SPECIFIED FILE. *)
5077 (*                                                               *)
5078 (*****************************************************************)
5079
5080 VAR INDEX : INCOREINDEX:
5081
5082 BEGIN (* GETLAST *)
5083
5084    IF (LINES.LAST <> NULLPOINTER)
5085       THEN
5086          BEGIN
5087             INDEX := LINES.LAST - 1:
5088
5089             WHILE (INCORETEXT [INDEX - 1] <> LM [2]) OR
5090                   (INCORETEXT [INDEX - 2] <> LM [1]) DO
5091                INDEX := INDEX - 1:
5092
5093             INDEX := INDEX - 2:
5094          END:
5095
5096    IF (LINES.LAST = NULLPOINTER)
5097       THEN
5098          BEGIN
5099             LEADINGLINES := NULLSEQ:
5100             LASTLINE := NULLSEQ:
5101          END
5102       ELSE
5103          BEGIN
5104
5105             IF (INDEX = LINES.FIRST)
5106                THEN
5107                   LEADINGLINES := NULLSEQ
5108                ELSE
5109                   BEGIN
5110                      LEADINGLINES.FIRST := LINES.FIRST:
5111                      LEADINGLINES.LAST  := INDEX - 1:
5112                   END:
5113
5114             LASTLINE.FIRST := INDEX:
5115             LASTLINE.LAST  := LINES.LAST:
5116          END:
5117
5118 END: (* GETLAST *)
5119
5120 (*27 *)
```

```
5121
5122 PROCEDURE GETTAIL(    LINES : INCOREPTRS;
5123                   VAR TAILLINE, TRAILINGLINES : INCOREPTRS );
5124
5125 (*****************************************************************)
5126 (*                                                               *)
5127 (* GETTAIL WILL RETURN THE REMAINING PORTION OF THE FIRST LINE.  *)
5128 (*                                                               *)
5129 (*****************************************************************)
5130
5131 VAR INDEX : INCOREINDEX;
5132
5133 BEGIN (* GETTAIL *)
5134
5135    IF (LINES.FIRST <> NULLPOINTER)
5136       THEN
5137          IF (INCORETEXT [LINES.FIRST] = LM [1]) AND
5138             (INCORETEXT [LINES.FIRST+1] = LM [2])
5139             THEN
5140                INDEX := NULLPOINTER
5141             ELSE
5142                BEGIN
5143                   INDEX := LINES.FIRST - 1;
5144
5145                   WHILE (INCORETEXT [INDEX + 1] <> RM [1]) OR
5146                         (INCORETEXT [INDEX + 2] <> RM [2]) DO
5147                      INDEX := INDEX + 1;
5148
5149                   INDEX := INDEX + 2;
5150                END;
5151
5152    IF (LINES.FIRST = NULLPOINTER)
5153       THEN
5154          BEGIN
5155             TAILLINE := NULLSEQ;
5156             TRAILINGLINES := NULLSEQ;
5157          END
5158       ELSE
5159          IF (INDEX = NULLPOINTER)
5160             THEN
5161                BEGIN
5162                   TAILLINE := NULLSEQ;
5163                   TRAILINGLINES := LINES;
5164                END
5165             ELSE
5166                BEGIN
5167
5168                   TAILLINE.FIRST := LINES.FIRST;
5169                   TAILLINE.LAST  := INDEX;
5170
5171                   IF (INDEX = LINES.LAST)
5172                      THEN
5173                         TRAILINGLINES := NULLSEQ
5174                      ELSE
5175                         BEGIN
5176                            TRAILINGLINES.FIRST := INDEX + 1;
5177                            TRAILINGLINES.LAST  := LINES.LAST;
5178                         END;
5179                END;
5180
5181 END; (* GETTAIL *)
5182
5183 (*97 *)
```

```
5184
5185 PROCEDURE GETHEAD(    LINES : INCOREPTRS:
5186                   VAR LEADINGLINES, HEADLINE : INCOREPTRS ):
5187
5188 (*****************************************************************)
5189 (*                                                               *)
5190 (* GETHEAD WILL RETURN THE BEGINNING PORTION OF THE LAST LINE.   *)
5191 (*                                                               *)
5192 (*****************************************************************)
5193
5194 VAR INDEX : INCOREINDEX:
5195
5196 BEGIN (* GETHEAD *)
5197
5198    IF (LINES.LAST <> NULLPOINTER)
5199       THEN
5200          IF (INCORETEXT [LINES.LAST] = RM [2]) AND
5201             (INCORETEXT [LINES.LAST-1] = RM [1])
5202             THEN
5203                INDEX := NULLPOINTER
5204             ELSE
5205                BEGIN
5206                   INDEX := LINES.LAST + 1:
5207
5208                   WHILE (INCORETEXT [INDEX - 1] <> LM [2]) OR
5209                         (INCORETEXT [INDEX - 2] <> LM [1]) DO
5210                      INDEX := INDEX - 1:
5211
5212                   INDEX := INDEX - 2:
5213                END:
5214
5215    IF (LINES.LAST = NULLPOINTER)
5216       THEN
5217          BEGIN
5218             LEADINGLINES := NULLSEQ:
5219             HEADLINE := NULLSEQ:
5220          END
5221       ELSE
5222          IF (INDEX = NULLPOINTER)
5223             THEN
5224                BEGIN
5225                   LEADINGLINES := LINES:
5226                   HEADLINE := NULLSEQ:
5227                END
5228             ELSE
5229                BEGIN
5230
5231                   IF (INDEX = LINES.FIRST)
5232                      THEN
5233                         LEADINGLINES := NULLSEQ
5234                      ELSE
5235                         BEGIN
5236                            LEADINGLINES.FIRST := LINES.FIRST:
5237                            LEADINGLINES.LAST  := INDEX - 1:
5238                         END:
5239
5240                   HEADLINE.FIRST := INDEX:
5241                   HEADLINE.LAST  := LINES.LAST:
5242                END:
5243
5244 END: (* GETHEAD *)
5245
5246 (*$7 *)
```

```
5247
5248 PROCEDURE LINEBALANCE( VAR HEAD, HIT, TAIL : INCOREPTRS );
5249
5250 (*************************************************************************)
5251 (*                                                                       *)
5252 (* LINEBALANCE WILL MODIFY THE HEAD, HIT AND TAIL SO THAT EACH           *)
5253 (* CONTAINS ONLY WHOLE LINES.                                            *)
5254 (*                                                                       *)
5255 (*************************************************************************)
5256
5257 VAR HEADLINE,
5258     LEADINGLINES,
5259
5260     TAILLINE,
5261     TRAILINGLINES : INCOREPTRS;
5262
5263 BEGIN (* LINEBALANCE *)
5264
5265    GETHEAD (HEAD, LEADINGLINES, HEADLINE);
5266
5267    HEAD := LEADINGLINES;
5268    APPEND (HEADLINE, HIT);
5269
5270    GETTAIL (TAIL, TAILLINE, TRAILINGLINES);
5271
5272    APPEND (TAILLINE, HIT);
5273    TAIL := TRAILINGLINES;
5274
5275 END; (* LINEBALANCE *)
5276
5277 (*$7 *)
```

```
5278
5279 PROCEDURE SHIFTTEXT(FLAG  : DIRECTION;
5280                     LINES : INCOREPTRS);
5281
5282 (*************************************************************************)
5283 (*                                                                       *)
5284 (* SHIFTTEXT WILL MOVE THE SPECIFIED LINES IN THE GIVEN DIRECTION        *)
5285 (* IN THE INCORETEXT BY MOVING THE SPECIFIED CHARACTERS.                 *)
5286 (*                                                                       *)
5287 (*************************************************************************)
5288
5289 VAR NEWBASE,
5290     OLDBASE : INCOREINDEX;
5291
5292     I,
5293     NUMCHARS : INTEGER;
5294
5295 BEGIN (* SHIFTTEXT *)
5296
5297    IF (LINES.FIRST <> NULLPOINTER) AND
5298       (HOLE.FIRST <> NULLPOINTER)
5299       THEN
5300          IF (FLAG = UP)
5301             THEN
5302                BEGIN
5303                   NEWBASE  := HOLE.FIRST;
5304                   OLDBASE  := LINES.FIRST;
5305                   NUMCHARS := LINES.LAST - LINES.FIRST;
5306
5307                   FOR I := 0 TO NUMCHARS DO
5308                      INCORETEXT[NEWBASE+I] := INCORETEXT[OLDBASE+I];
5309
5310                   HOLE.LAST := LINES.LAST;
5311
5312                   LINES.FIRST := NEWBASE;
5313                   LINES.LAST  := NEWBASE + NUMCHARS;
5314
5315                   HOLE.FIRST := LINES.LAST + 1;
5316
5317                   APPEND (LINES, UPPERSEQ);
5318                END
5319             ELSE
5320                BEGIN
5321                   NEWBASE  := HOLE.LAST;
5322                   OLDBASE  := LINES.LAST;
5323                   NUMCHARS := LINES.LAST - LINES.FIRST;
5324
5325                   FOR I := 0 TO NUMCHARS DO
5326                      INCORETEXT[NEWBASE-I] := INCORETEXT[OLDBASE-I];
5327
5328                   HOLE.FIRST := LINES.FIRST;
5329
5330                   LINES.FIRST := NEWBASE - NUMCHARS;
5331                   LINES.LAST  := NEWBASE;
5332
5333                   HOLE.LAST := LINES.FIRST - 1;
5334
5335                   APPEND (LINES, LOWERSEQ);
5336                END;
5337
5338 END; (* SHIFTTEXT *)
5339
5340 (*$7 *)
```

```
5341
5342 PROCEDURE GETNEWLINES:
5343
5344 (*************************************************************************)
5345 (*                                                                       *)
5346 (* GETNEWLINES WILL GET THE NEW LINES FROM A FILE, THE TERMINAL OR       *)
5347 (* THE INPUT LINE AND PLACE THEM ONTO THE SCRATCH FILE.                  *)
5348 (*                                                                       *)
5349 (*************************************************************************)
5350
5351 VAR I, J : STRINGINDEX:
5352
5353     LINE : STRING:
5354
5355     INDEX,
5356     LENGTH : STRINGINDEX:
5357
5358     FOUND,
5359     FINISHED : BOOLEAN:
5360
5361 BEGIN (* GETNEWLINES *)
5362
5363    VALIDINFO := FALSE:
5364
5365    IF (REQUEST.FILENAME = INPUTFILE)
5366       THEN
5367          BEGIN
5368
5369             SHOWMESSAGE( ENTER ):
5370             FINISHED := FALSE:
5371
5372             REPEAT
5373
5374                GETLINE( ADDPROMPT, LINE, LENGTH ):
5375
5376                IF INTERRUPT
5377                   THEN
5378                      FINISHED := TRUE
5379                   ELSE
5380                      BEGIN
5381                         IF (LENGTH > NULLSTRLEN)
5382                            THEN
5383                               FOR I := 1 TO LENGTH DO
5384                                  WRITE (SCRATCH, LINE[I]):
5385
5386                         WRITELN (SCRATCH):
5387
5388                         VALIDINFO := TRUE:
5389                      END:
5390
5391             UNTIL FINISHED:
5392
5393             SHOWMESSAGE( ENDTEXT ):
5394
5395          END:
5396
5397
5398    INDEX := 0:
5399
5400    WITH PATTERN DO
5401       IF (REQUEST.FILENAME = EMPTYFILE )
5402          THEN
5403             BEGIN
```

```
5404                  WHILE (INDEX < PATLEN) DO
5405                     BEGIN
5406
5407                        FOUND := FALSE:
5408
5409                        IF (INDEX < PATLEN - 1)
5410                           THEN
5411                              IF (CHR (INDEX+1) = LM (1)) AND
5412                                 (CHR (INDEX+2) = LM (2))
5413                                 THEN
5414                                  INDEX := INDEX + 2:
5415
5416                        WHILE (INDEX < PATLEN) AND NOT(FOUND) DO
5417                           BEGIN
5418                              IF (INDEX < PATLEN - 1)
5419                                 THEN
5420                                  IF (CHR (INDEX+1) = RM (1)) AND
5421                                     (CHR (INDEX+2) = RM (2))
5422                                   THEN
5423                                    FOUND := TRUE:
5424
5425                              IF NOT(FOUND)
5426                                 THEN
5427                                  BEGIN
5428                                   WRITE (SCRATCH, CHR (INDEX+1)):
5429                                   INDEX := INDEX + 1:
5430                                  END:
5431                           END:
5432
5433                        IF (INDEX < PATLEN - 1)
5434                           THEN
5435                              IF (CHR (INDEX+1) = RM (1)) AND
5436                                 (CHR (INDEX+2) = RM (2))
5437                                 THEN
5438                                  BEGIN
5439                                   WRITELN (SCRATCH):
5440                                   INDEX := INDEX + 2:
5441                                  END:
5442
5443                     END:
5444
5445                  WRITELN (SCRATCH):
5446                  VALIDINFO := TRUE:
5447               END:
5448
5449
5450     IF (REQUEST.FILENAME <> EMPTYFILE) AND
5451        (REQUEST.FILENAME <> INPUTFILE)
5452        THEN
5453           SETUP( REQUEST.FILENAME ):
5454
5455
5456     IF NOT(ERROR)
5457        THEN
5458           READFILE:
5459
5460     REWRITE (SCRATCH):
5461     VALIDINFO := FALSE:
5462
5463  END: (* GETNEWLINES *)
5464
5465  (*$7 *)
```

```
5466
5467 PROCEDURE PUTLINECHANGE;
5468
5469 (*****************************************************************)
5470 (*                                                               *)
5471 (* PUTLINECHANGE WILL MAKE THE MODIFICATIONS INDICATED IN NEW TEXT.   *)
5472 (*                                                               *)
5473 (*****************************************************************)
5474
5475 VAR I : STRINGINDEX;
5476
5477     J : INCOREINDEX;
5478
5479     NUMELL,
5480     PATINDEX,
5481     REPINDEX : STRINGINDEX;
5482
5483     LENGTH,
5484     ELLSIZE,
5485     OFFSET,
5486     BASE : INCOREINDEX;
5487
5488     HEADLINE,
5489     REMAININGLINES,
5490
5491     LEADINGLINES,
5492     TAILLINE,
5493
5494     LINES : INCOREPTRS;
5495
5496     PATFOUND,
5497     REPFOUND,
5498
5499     FINISHED,
5500     CHECKIN,
5501     CHECKOUT : BOOLEAN;
5502
5503 BEGIN (* PUTLINECHANGE *)
5504
5505    FINISHED := FALSE;
5506    PATINDEX := 0;
5507    REPINDEX := 0;
5508
5509    WHILE NOT(FINISHED) DO
5510       BEGIN
5511          PATFOUND := FALSE;
5512          REPFOUND := FALSE;
5513
5514          WHILE (PATINDEX < PATLEN) AND NOT(PATFOUND) DO
5515             BEGIN
5516                PATINDEX := PATINDEX + 1;
5517
5518                IF (PATTERN.SYM [PATINDEX] = WILDCARD)
5519                   THEN
5520                      PATFOUND := TRUE;
5521             END;
5522
5523          WHILE (REPINDEX < REPLEN) AND NOT(REPFOUND) DO
5524             BEGIN
5525                REPINDEX := REPINDEX + 1;
5526
5527                IF (REPLACEMENT.SYM [REPINDEX] = WILDCARD)
5528                   THEN
```

```
5529                        REPFOUND := TRUE;
5530                END;
5531
5532            IF (PATFOUND AND REPFOUND)
5533                THEN
5534                    BEGIN
5535                        REPLACEMENT.JOK[REPINDEX] := PATTERN.JOK[PATINDEX];
5536                        REPLACEMENT.CHR[REPINDEX] := PATTERN.JOK[PATINDEX];
5537                    END;
5538
5539            IF (PATINDEX = PATLEN) OR
5540                (REPINDEX = REPLEN)
5541                THEN
5542                    FINISHED := TRUE;
5543        END;
5544
5545    FINISHED := FALSE;
5546    PATINDEX := 0;
5547    REPINDEX := 0;
5548    ELLSIZE := 0;
5549    NUMELL := 0;
5550
5551    WHILE NOT(FINISHED) DO
5552        BEGIN
5553            PATFOUND := FALSE;
5554            REPFOUND := FALSE;
5555
5556            WHILE (PATINDEX < PATLEN) AND NOT(PATFOUND) DO
5557                BEGIN
5558                    PATINDEX := PATINDEX + 1;
5559
5560                    IF (PATTERN.SYM [PATINDEX] = FREESTRING)
5561                        THEN
5562                            PATFOUND := TRUE;
5563                END;
5564
5565            WHILE (REPINDEX < REPLEN) AND NOT(REPFOUND) DO
5566                BEGIN
5567                    REPINDEX := REPINDEX + 1;
5568
5569                    IF (REPLACEMENT.SYM [REPINDEX] = FREESTRING)
5570                        THEN
5571                            REPFOUND := TRUE;
5572                END;
5573
5574            IF (PATFOUND AND REPFOUND)
5575                THEN
5576                    BEGIN
5577                        REPLACEMENT.ELL[REPINDEX] := PATTERN.ELL[PATINDEX];
5578                        ELLSIZE := ELLSIZE +
5579                                   REPLACEMENT.ELL[REPINDEX].LAST -
5580                                   REPLACEMENT.ELL[REPINDEX].FIRST + 1;
5581                        NUMELL := NUMELL + 1;
5582                    END;
5583
5584            IF (PATINDEX = PATLEN) OR
5585                (REPINDEX = REPLEN)
5586                THEN
5587                    FINISHED := TRUE;
5588        END;
5589
5590    CHECKIN  := FALSE;
5591    CHECKOUT := FALSE;
5592
```

```
5593     WITH REPLACEMENT DO
5594        IF (REPLEN = 1)
5595           THEN
5596              BEGIN
5597
5598                 IF (CHR [1] = OUTMARKER)
5599                    THEN
5600                       CHECKIN := TRUE;
5601
5602                 IF (CHR [1] = OUTMARKER)
5603                    THEN
5604                       CHECKOUT := TRUE;
5605
5606              END
5607           ELSE
5608              BEGIN
5609
5610                 IF (CHR [1] = LM [1]) AND
5611                    (CHR [2] = LM [2])
5612                    THEN
5613                       CHECKIN := TRUE;
5614
5615                 IF (CHR [REPLEN-1] = RM [1]) AND
5616                    (CHR [REPLEN] = RM [2])
5617                    THEN
5618                       CHECKOUT := TRUE;
5619
5620              END;
5621
5622     IF (HOLE.FIRST = NULLPOINTER)
5623        THEN
5624           BASE := NULLPOINTER
5625        ELSE
5626           BASE := HOLE.FIRST - 1;
5627
5628     LENGTH := REPLEN + ELLSIZE - NUMELL;
5629
5630     WITH REPLACEMENT DO
5631        IF (LENGTH + 2 > HOLE.LAST - BASE) OR
5632           (HOLE.FIRST = NULLPOINTER)
5633           THEN
5634              SHOWERROR( ERRNOROOM )
5635           ELSE
5636              BEGIN
5637
5638                 OFFSET := 0;
5639
5640                 IF (CHECKIN)
5641                    THEN
5642                       BEGIN
5643                          GETHEAD (UPPERSEQ, HEADLINE,
5644                                   REMAINGLINES);
5645
5646                          IF (HEADLINE.FIRST = HEADLINE.LAST - 1)
5647                             THEN
5648                                BEGIN
5649                                 INCORETEXT [BASE + 1] := BLANK;
5650                                 BASE := BASE + 1;
5651                                 OFFSET := OFFSET + 1;
5652                                END;
5653                       END;
5654
5655                 FOR I := 1 TO REPLEN DO
5656                    CASE SYM [I] OF
```

```
5657
5658                    FREESTRING : IF (ELL [I].FIRST <> NULLPOINTER)
5659                                 THEN
5660                                  FOR J := ELL [I].FIRST TO
5661                                      ELL [I].LAST DO
5662                                   BEGIN
5663                                    INCORETEXT[BASE+I]
5664                                    := TEMPINCORETEXT[J]:
5665                                    IF (J <> ELL [I].LAST)
5666                                     THEN
5667                                      BEGIN
5668                                       BASE := BASE + 1:
5669                                       OFFSET := OFFSET + 1:
5670                                      END:
5671                                   END:
5672
5673                       CONSTANT    : INCORETEXT [BASE + I] := CHR [I]:
5674
5675                       WILDCARD    : INCORETEXT [BASE + I] := JOK [I]:
5676
5677                    END: (* CASE *)
5678
5679                 IF (CHECKOUT)
5680                    THEN
5681                       BEGIN
5682                          GETTAIL (LOWERSEQ, LEADINGLINES,
5683                                   TAILLINE):
5684
5685                          IF (TAILLINE.FIRST = TAILLINE.LAST - 1)
5686                             THEN
5687                                BEGIN
5688                                 BASE := BASE + 1:
5689                                 OFFSET := OFFSET + 1:
5690                                 INCORETEXT[BASE + LENGTH] := BLANK:
5691                                END:
5692                       END:
5693
5694                 LINES.FIRST := BASE - OFFSET + 1:
5695                 LINES.LAST  := BASE - OFFSET + LENGTH:
5696
5697                 IF (LINES.LAST = HOLE.LAST)
5698                    THEN
5699                       HOLE := NULLSEQ
5700                    ELSE
5701                       HOLE.FIRST := LINES.LAST + 1:
5702
5703                 APPEND (LINES, UPPERSEQ):
5704
5705              END:
5706
5707 END: (* PUTLINECHANGE *)
5708
5709 (*$7 *)
```

```
5710
5711 PROCEDURE TAKEBACKREQUEST;
5712
5713 (*****************************************************************************)
5714 (*                                                                           *)
5715 (* TAKEBACKREQUEST WILL RESTORE THE STATE TO WHAT IT WAS BEFORE THE          *)
5716 (* CURRENT REQUEST WAS ATTEMPTED (INTERNAL UNDO).                            *)
5717 (*                                                                           *)
5718 (*****************************************************************************)
5719
5720 BEGIN (* TAKEBACKREQUEST *)
5721
5722    REWRITE (SCRATCH);
5723    VALIDINFO := FALSE;
5724
5725    SHOWMESSAGE( NOTDONE );
5726
5727    IF REQUEST.NAME IN [FORWARD..ASSUME]
5728       THEN
5729          BEGIN
5730
5731             TOPSEQ      := TEMPTOPSEQ;
5732             UPPERSEQ    := TEMPUPPERSEQ;
5733             HOLE        := TEMPHOLE;
5734             CURRENTLINE := TEMPCURRENTLINE;
5735             LOWERSEQ    := TEMPLOWERSEQ;
5736             BOTTOMSEQ   := TEMPBOTTOMSEQ;
5737
5738             INCORETEXT  := TEMPINCORETEXT;
5739
5740
5741             FILECHANGES := TEMPFILECHANGES;
5742             FILESTATUS  := TEMPFILESTATUS;
5743
5744             CURFILENAME := TEMPCURFILENAME;
5745
5746             CURALPHABET := TEMPCURALPHABET;
5747
5748             MARGINSYM   := TEMPMARGINSYM;
5749             JOKERSYM    := TEMPJOKERSYM;
5750             ELLIPSISSYM := TEMPELLIPSISSYM;
5751
5752             IF (CURFILENAME <> EMPTYFILE)
5753                THEN
5754                   SHOWVALUE( VALCURLINE );
5755
5756          END;
5757
5758    ENDOFLINE := TRUE;
5759    SHOWLINE := FALSE;
5760
5761 END; (* TAKEBACKREQUEST *)
5762
5763 (*$Z *)
```

```
5764
5765 PROCEDURE SEMFORWARD:
5766
5767 (*************************************************************************)
5768 (*                                                                       *)
5769 (* SEMFORWARD PERFORMS THE SEMANTICS OF THE FORWARD REQUEST. THE         *)
5770 (* CURRENT LINE WILL BE MOVED TO THE LAST LINE OF THE SPECIFIED          *)
5771 (* TEXT.                                                                 *)
5772 (*                                                                       *)
5773 (*************************************************************************)
5774
5775 VAR SEARCHLINES,
5776     HEAD,
5777     HIT,
5778     TAIL,
5779
5780     LASTLINE,
5781     LEADINGLINES : INCOREPTRS:
5782
5783     ANSWER : KEYSYMBOL:
5784
5785 BEGIN (* SEMFORWARD *)
5786
5787    MATCHNUM := REQUEST.REPEATER:
5788
5789    SEARCHLINES := LOWERSEQ:
5790
5791    REPEAT
5792
5793       CORRFILEFRAG(SEARCHLINES, HEAD, HIT, TAIL):
5794
5795       LINEBALANCE (HEAD, HIT, TAIL):
5796
5797       IF (HIT.FIRST <> NULLPOINTER)
5798          THEN
5799             BEGIN
5800
5801                GETLAST (HIT, LEADINGLINES, LASTLINE):
5802
5803                SHIFTTEXT (UP, CURRENTLINE):
5804                SHIFTTEXT (UP, HEAD):
5805                SHIFTTEXT (UP, LEADINGLINES):
5806
5807                CURRENTLINE := LASTLINE:
5808                LOWERSEQ    := TAIL:
5809
5810                SEARCHLINES := TAIL:
5811
5812                MATCHNUM := MATCHNUM - 1:
5813
5814             END
5815          ELSE
5816             IF (REQUEST.REPEATER <> REPEATALL) OR
5817                (MATCHNUM = REQUEST.REPEATER)
5818                THEN
5819                   BEGIN
5820
5821                      SHOWERROR( ERRNOTFOUND ):
5822
5823                      IF (MATCHNUM <> REQUEST.REPEATER)
5824                         THEN
5825                            SECURITYCHECK( DOREQUEST, ANSWER )
5826                         ELSE
```

```
5827                                ANSWER := NO;
5828
5829                            IF (ANSWER = NO)
5830                               THEN
5831                                  TAKEBACKREQUEST;
5832
5833                         END;
5834
5835     UNTIL DISABLE OR (MATCHNUM = 0) OR (HIT.FIRST = NULLPOINTER);
5836
5837     IF NOT(DISABLE)
5838        THEN
5839           IF (ENDOFLINE)
5840              THEN
5841                 BEGIN
5842                    WRITEFILE (CURRENTLINE);
5843                    SHOWLISTING;
5844                 END;
5845
5846 END; (* SEMFORWARD *)
5847
5848 (*$7 *)
```

```
5849
5850 PROCEDURE SEMBACKWARD;
5851
5852 (*************************************************************************)
5853 (*                                                                       *)
5854 (* SEMBACKWARD PERFORMS THE SEMANTICS OF THE BACKWARD REQUEST. THE       *)
5855 (* CURRENT LINE WILL BE MOVED TO THE FIRST LINE OF THE SPECIFIED         *)
5856 (* TEXT.                                                                 *)
5857 (*                                                                       *)
5858 (*************************************************************************)
5859
5860 VAR SEARCHLINES,
5861     HEAD,
5862     HIT,
5863     TAIL,
5864
5865     LINES,
5866
5867     FIRSTLINE,
5868     REMAININGLINES : INCOREPTRS;
5869
5870     ANSWER : KEYSYMBOL;
5871
5872 BEGIN (* SEMBACKWARD *)
5873
5874    MATCHNUM := REQUEST.REPEATER;
5875
5876    SEARCHLINES := UPPERSEQ;
5877
5878    REPEAT
5879
5880       REVERSE (SEARCHLINES);
5881
5882       COPYFILEFRAG(SEARCHLINES, HEAD, HIT, TAIL);
5883
5884       REVERSE (HEAD);
5885       REVERSE (HIT);
5886       REVERSE (TAIL);
5887
5888       LINEBALANCE (TAIL, HIT, HEAD);
5889
5890       IF (HIT.FIRST <> NULLPOINTER)
5891          THEN
5892             BEGIN
5893
5894                APPEND (CURRENTLINE, LOWERSEQ);
5895
5896                SHIFTTEXT (DOWN, HEAD);
5897                LINES := HIT;
5898                SHIFTTEXT (DOWN, LINES);
5899
5900                GETFIRST (LOWERSEQ, FIRSTLINE, REMAININGLINES);
5901
5902                CURRENTLINE := FIRSTLINE;
5903                LOWERSEQ    := REMAININGLINES;
5904
5905                UPPERSEQ := TAIL;
5906
5907                SEARCHLINES := TAIL;
5908
5909                MATCHNUM := MATCHNUM - 1;
5910
5911             END
```

```
5912            ELSE
5913               IF (REQUEST.REPEATER <> REPEATALL) OR
5914                  (MATCHNUM = REQUEST.REPEATER)
5915                  THEN
5916                     BEGIN
5917
5918                        SHOWERROR( ERRNOTFOUND ):
5919
5920                        IF (MATCHNUM <> REQUEST.REPEATER)
5921                           THEN
5922                              SECURITYCHECK( DOREQUEST, ANSWER )
5923                           ELSE
5924                              ANSWER := NO:
5925
5926                        IF (ANSWER = NO)
5927                           THEN
5928                              TAKEBACKREQUEST:
5929
5930                     END:
5931
5932     UNTIL DISABLE OR (MATCHNUM = 0) OR (HIT.FIRST = NULLPOINTER):
5933
5934     IF NOT(DISABLE)
5935        THEN
5936           IF (ENDOFLINE)
5937              THEN
5938                 BEGIN
5939                    WRITEFILE (CURRENTLINE);
5940                    SHOWLISTING:
5941                 END:
5942
5943 END: (* SEMBACKWARD *)
5944
5945 (*$Z *)
```

```
5946
5947 PROCEDURE SEMLIST;
5948
5949 (*****************************************************************)
5950 (*                                                               *)
5951 (* SEMLIST PERFORMS THE SEMANTICS OF THE LIST REQUEST. THE CURRENT    *)
5952 (* LINE WILL BE MOVED TO THE LAST LINE OF THE SPECIFIED TEXT.         *)
5953 (*                                                               *)
5954 (*****************************************************************)
5955
5956 VAR SEARCHLINES,
5957     HEAD,
5958     HIT,
5959     TAIL,
5960
5961     LASTLINE,
5962     LEADINGLINES : INCOREPTRS;
5963
5964     ANSWER : KEYSYMBOL;
5965
5966 BEGIN (* SEMLIST *)
5967
5968    LINESFOUND := 0;
5969
5970    MATCHNUM := REQUEST.REPEATER;
5971
5972    APPEND (CURRENTLINE, LOWERSEQ);
5973
5974    SEARCHLINES := LOWERSEQ;
5975
5976    REPEAT
5977
5978       CORRFILEFRAG(SEARCHLINES, HEAD, HIT, TAIL);
5979
5980       LINEBALANCE (HEAD, HIT, TAIL);
5981
5982       IF (HIT.FIRST <> NULLPOINTER)
5983          THEN
5984             BEGIN
5985
5986                REWRITE (SCRATCH);
5987                VALIDINFO := FALSE;
5988                WRITEFILE (HIT);
5989                SHOWLISTING;
5990
5991                SHOWLINE := FALSE;
5992
5993                GETLAST (HIT, LEADINGLINES, LASTLINE);
5994
5995                SHIFTTEXT (UP, CURRENTLINE);
5996                SHIFTTEXT (UP, HEAD);
5997                SHIFTTEXT (UP, LEADINGLINES);
5998
5999                CURRENTLINE := LASTLINE;
6000                LOWERSEQ    := TAIL;
6001
6002                SEARCHLINES := TAIL;
6003
6004                MATCHNUM := MATCHNUM - 1;
6005
6006             END
6007          ELSE
6008             IF (REQUEST.REPEATER <> REPEATALL) OR
```

```
6009                    (MATCHNUM = REQUEST.REPEATER)
6010                    THEN
6011                       BEGIN
6012
6013                          SHOWERROR( ERRNOTFOUND ):
6014
6015                          IF (MATCHNUM <> REQUEST.REPEATER)
6016                             THEN
6017                                SECURITYCHECK( DOREQUEST, ANSWER )
6018                             ELSE
6019                                ANSWER := NO:
6020
6021                          IF (ANSWER = NO)
6022                             THEN
6023                                BEGIN
6024                                   TAKEBACKREQUEST:
6025                                   REWRITE (SCRATCH):
6026                                   VALIDINFO := FALSE:
6027                                   WRITEFILE (CURRENTLINE):
6028                                   SHOWLISTING:
6029                                END;
6030
6031                       END:
6032
6033
6034     UNTIL DISABLE OR (MATCHNUM = 0) OR (HIT.FIRST = NULLPOINTER):
6035
6036     SHOWLINE := TRUE:
6037
6038 END: (* SEMLIST *)
6039
6040 (*$Z *)
```

```
6041
6042 PROCEDURE SEMDELETE;
6043
6044 (*****************************************************************)
6045 (*                                                               *)
6046 (* SEMDELETE PERFORMS THE SEMANTICS OF THE DELETE REQUEST. ALL   *)
6047 (* LINES CONTAINING OCCURRENCES OF THE SPECIFIED TEXT WILL BE     *)
6048 (* REMOVED. THE CURRENT LINE WILL BE MOVED TO THE LINE FOLLOWING T*)
6049 (* THE LAST LINE OF THE SPECIFIED TEXT.                          *)
6050 (*                                                               *)
6051 (*****************************************************************)
6052
6053 VAR SEARCHLINES,
6054     HEAD,
6055     HIT,
6056     TAIL,
6057
6058     LINES,
6059
6060     FIRSTLINE,
6061     REMAININGLINES : INCOREPTRS;
6062
6063     ANSWER : KEYSYMBOL;
6064
6065 BEGIN (* SEMDELETE *)
6066
6067    LINESFOUND := 0;
6068
6069    MATCHNUM := REQUEST.REPEATER;
6070
6071    APPEND (CURRENTLINE, LOWERSEQ);
6072
6073    SEARCHLINES := LOWERSEQ;
6074
6075    REPEAT
6076
6077       CORRFILEFRAG(SEARCHLINES, HEAD, HIT, TAIL);
6078
6079       LINEBALANCE (HEAD, HIT, TAIL);
6080
6081       IF (HIT.FIRST <> NULLPOINTER)
6082          THEN
6083             BEGIN
6084
6085                WRITEFILE (HIT);
6086
6087                SHIFTTEXT (UP, HEAD);
6088                LINES := HIT;
6089                APPEND (LINES, HOLE);
6090
6091                GETFIRST (TAIL, FIRSTLINE, REMAININGLINES);
6092
6093                CURRENTLINE := FIRSTLINE;
6094                LOWERSEQ    := REMAININGLINES;
6095
6096                SEARCHLINES := TAIL;
6097
6098                MATCHNUM := MATCHNUM - 1;
6099
6100             END
6101          ELSE
6102             IF (REQUEST.REPEATER <> REPEATALL) OR
6103                (MATCHNUM = REQUEST.REPEATER)
```

```
6104                    THEN
6105                       BEGIN
6106
6107                          SHOWERROR( ERRNOTFOUND ):
6108
6109                          IF (MATCHNUM <> REQUEST.REPEATER)
6110                             THEN
6111                                SECURITYCHECK( DOREQUEST, ANSWER )
6112                             ELSE
6113                                ANSWER := NO:
6114
6115                          IF (ANSWER = NO)
6116                             THEN
6117                                TAKEBACKREQUEST:
6118
6119                       END:
6120
6121     UNTIL DISABLE OR (MATCHNUM = 0) OR (HIT.FIRST = NULLPOINTER):
6122
6123     IF NOT(DISABLE)
6124        THEN
6125           BEGIN
6126              IF NOT(ERROR)
6127                 THEN
6128                    IF (LINESFOUND <> 0) AND
6129                       (REQUEST.TEXTLEN <> NULLSTRLEN)
6130                       THEN
6131                          SHOWMESSAGE( NUMLINES ):
6132
6133              IF (ENDOFLINE)
6134                 THEN
6135                    BEGIN
6136                       REWRITE (SCRATCH):
6137                       VALIDINFO := FALSE:
6138                       WRITEFILE (CURRENTLINE):
6139                       SHOWLISTING:
6140                    END:
6141           END:
6142
6143  END: (* SEMDELETE *)
6144
6145  (*$Z *)
```

```
6146
6147 PROCEDURE SEMTRANSFER;
6148
6149 (*************************************************************************)
6150 (*                                                                       *)
6151 (* SEMTRANSFER PERFORMS THE SEMANTICS OF THE TRANSFER REQUEST. THE       *)
6152 (* INDICATED TEXT IS REMOVED FROM THE CURRENT FILE AND PLACED INTO       *)
6153 (* THE NAMED FILE. IF THE FILE ALREADY EXISTED, A SECURITY CHECK IS      *)
6154 (* PERFORMED. THE CURRENT LINE IS MOVED TO THE LINE FOLLOWING THE        *)
6155 (* INDICATED TEXT.                                                       *)
6156 (*                                                                       *)
6157 (*************************************************************************)
6158
6159 VAR SEARCHLINES,
6160     HEAD,
6161     HIT,
6162     TAIL,
6163     LINES,
6164     FIRSTLINE,
6165     REMAININGLINES : INCOREPTRS;
6166
6167     ANSWER : KEYSYMBOL;
6168
6169 BEGIN (* SEMTRANSFER *)
6170
6171    LINESFOUND := 0;
6172
6173    MATCHNUM := REQUEST.REPEATER;
6174
6175    APPEND (CURRENTLINE, LOWERSEQ);
6176
6177    SEARCHLINES := LOWERSEQ;
6178
6179    REPEAT
6180
6181       CORRFILEFRAG(SEARCHLINES, HEAD, HIT, TAIL);
6182
6183       LINEBALANCE (HEAD, HIT, TAIL);
6184
6185       IF (HIT.FIRST <> NULLPOINTER)
6186          THEN
6187             BEGIN
6188
6189                WRITEFILE (HIT);
6190
6191                GETFIRST (TAIL, FIRSTLINE, REMAININGLINES);
6192
6193                SHIFTTEXT (UP, HEAD);
6194                LINES := HIT;
6195                APPEND (LINES, HOLE);
6196
6197                CURRENTLINE := FIRSTLINE;
6198                LOWERSEQ    := REMAININGLINES;
6199
6200                SEARCHLINES := TAIL;
6201
6202                MATCHNUM := MATCHNUM - 1;
6203
6204             END
6205          ELSE
6206             IF (REQUEST.REPEATER <> REPEATALL) OR
6207                (MATCHNUM = REQUEST.REPEATER)
6208                THEN
```

```
6209                    BEGIN
6210
6211                       SHOWERROR( ERRNOTFOUND ):
6212
6213                       IF (MATCHNUM <> REQUEST.REPEATER)
6214                          THEN
6215                             SECURITYCHECK( DOREQUEST, ANSWER )
6216                          ELSE
6217                             ANSWER := NO:
6218
6219                       IF (ANSWER = NO)
6220                          THEN
6221                             TAKEBACKREQUEST:
6222
6223                    END:
6224
6225    UNTIL DISABLE OR (MATCHNUM = 0) OR (HIT.FIRST = NULLPOINTER):
6226
6227    IF NOT(DISABLE)
6228       THEN
6229          BEGIN
6230             IF NOT(ERROR)
6231                THEN
6232                   BEGIN
6233
6234                      IF HAVEFILE( REQUEST.FILENAME )
6235                         THEN
6236                            SECURITYCHECK( OVERWRITEFILE, ANSWER )
6237                         ELSE
6238                            BEGIN
6239                               SHOWMESSAGE( NEWFILE ):
6240                               ANSWER := YES:
6241                            END:
6242
6243                      IF (ANSWER = NO)
6244                         THEN
6245                            TAKEBACKREQUEST
6246                         ELSE
6247                            BEGIN
6248                               PUTINTO (REQUEST.FILENAME):
6249
6250                               IF ERROR
6251                                THEN
6252                                 TAKEBACKREQUEST
6253                                ELSE
6254                                 IF (LINESFOUND <> 0) AND
6255                                    (REQUEST.TEXTLEN <> NULLSTRLEN)
6256                                  THEN
6257                                   SHOWMESSAGE( NUMLINES ):
6258                            END:
6259                   END:
6260
6261             IF (ENDOFLINE)
6262                THEN
6263                   BEGIN
6264                      REWRITE (SCRATCH):
6265                      VALIDINFO := FALSE:
6266                      WRITEFILE (CURRENTLINE):
6267                      SHOWLISTING:
6268                   END;
6269          END:
6270
6271 END: (* SEMTRANSFER *)
6272
```

```
6274
6275 PROCEDURE SEMCOPY;
6276
6277 (*****************************************************************)
6278 (*                                                               *)
6279 (* SEMCOPY PERFORMS THE SEMANTICS OF THE COPY REQUEST. A COPY OF THE  *)
6280 (* INDICATED TEXT IS PLACED INTO THE SPECIFIED FILE. IF THE FILE     *)
6281 (* ALREADY EXISTED, A SECURITY CHECK IS PERFORMED. THE CURRENT LINE I *)
6282 (* MOVED TO THE LAST LINE OF THE INDICATED TEXT.                     *)
6283 (*                                                               *)
6284 (*****************************************************************)
6285
6286 VAR SEARCHLINES,
6287     HEAD,
6288     HIT,
6289     TAIL,
6290
6291     LASTLINE,
6292     LEADINGLINES : INCOREPTRS;
6293
6294     ANSWER : KEYSYMBOL;
6295
6296 BEGIN (* SEMCOPY *)
6297
6298    LINESFOUND := 0;
6299
6300    MATCHNUM := REQUEST.REPEATER;
6301
6302    APPEND (CURRENTLINE, LOWERSEQ);
6303
6304    SEARCHLINES := LOWERSEQ;
6305
6306    REPEAT
6307
6308       COPYFILEFRAG(SEARCHLINES, HEAD, HIT, TAIL);
6309
6310       LINEBALANCE (HEAD, HIT, TAIL);
6311
6312       IF (HIT.FIRST <> NULLPOINTER)
6313          THEN
6314             BEGIN
6315
6316                WRITEFILE (HIT);
6317
6318                GETLAST (HIT, LEADINGLINES, LASTLINE);
6319
6320                SHIFTTEXT (UP, CURRENTLINE);
6321                SHIFTTEXT (UP, HEAD);
6322                SHIFTTEXT (UP, LEADINGLINES);
6323
6324                CURRENTLINE := LASTLINE;
6325                LOWERSEQ := TAIL;
6326
6327                SEARCHLINES := TAIL;
6328
6329                MATCHNUM := MATCHNUM - 1;
6330
6331             END
6332          ELSE
6333             IF (REQUEST.REPEATER <> REPEATALL) OR
6334                (MATCHNUM = REQUEST.REPEATER)
6335                THEN
6336                   BEGIN
```

```
6337
6338                        SHOWERROR( ERRNOTFOUND ):
6339
6340                        IF (MATCHNUM <> REQUEST.REPEATER)
6341                           THEN
6342                              SECURITYCHECK( DOREQUEST, ANSWER )
6343                           ELSE
6344                              ANSWER := NO:
6345
6346                        IF (ANSWER = NO)
6347                           THEN
6348                              TAKEBACKREQUEST:
6349
6350                     END:
6351
6352     UNTIL DISABLE OR (MATCHNUM = 0) OR (HIT.FIRST = NULLPOINTER):
6353
6354     IF NOT(DISABLE)
6355        THEN
6356           BEGIN
6357              IF NOT(ERROR)
6358                 THEN
6359                    BEGIN
6360
6361                       IF HAVEFILE( REQUEST.FILENAME )
6362                          THEN
6363                             SECURITYCHECK( OVERWRITEFILE, ANSWER )
6364                          ELSE
6365                             BEGIN
6366                                SHOWMESSAGE( NEWFILE ):
6367                                ANSWER := YES:
6368                             END:
6369
6370                       IF (ANSWER = NO)
6371                          THEN
6372                             TAKEBACKREQUEST
6373                          ELSE
6374                             BEGIN
6375                                PUTINTO (REQUEST.FILENAME);
6376
6377                                IF ERROR
6378                                 THEN
6379                                  TAKEBACKREQUEST
6380                                 ELSE
6381                                  IF (LINESFOUND <> 0) AND
6382                                     (REQUEST.TEXTLEN <> NULLSTRLEN)
6383                                   THEN
6384                                    SHOWMESSAGE( NUMLINES ):
6385                             END;
6386                    END:
6387
6388                 IF (ENDOFLINE)
6389                    THEN
6390                       BEGIN
6391                          REWRITE (SCRATCH):
6392                          VALIDINFO := FALSE:
6393                          WRITEFILE (CURRENTLINE):
6394                          SHOWLISTING:
6395                       END:
6396              END:
6397
6398 END: (* SEMCOPY *)
6399
6400 (*$Z *)
```

```
6401
6402 PROCEDURE SEMMAKE;
6403
6404 (*************************************************************************)
6405 (*                                                                       *)
6406 (* SEMMAKE PERFORMS THE SEMANTICS OF THE MAKE REQUEST. THE CHANGES       *)
6407 (* WILL BE MADE TO THE SPECIFIED LINES. THE CURRENT LINE WILL BE         *)
6408 (* MOVED TO THE LAST LINE OF THE MODIFIED LINES.                         *)
6409 (*                                                                       *)
6410 (*************************************************************************)
6411
6412 VAR SEARCHLINES,
6413     HEAD,
6414     HIT,
6415     TAIL,
6416
6417     LINES,
6418
6419     TAILLINE,
6420
6421     FIRSTLINE,
6422     REMAININGLINES,
6423
6424     LASTLINE,
6425     LEADINGLINES : INCOREPTRS;
6426
6427     ANSWER : KEYSYMBOL;
6428
6429     NOTENOUGH : BOOLEAN;
6430
6431 BEGIN (* SEMMAKE *)
6432
6433    NOTENOUGH := FALSE;
6434
6435    MATCHNUM := REQUEST.REPEATER;
6436
6437    APPEND (CURRENTLINE, LOWERSEQ);
6438
6439    SEARCHLINES := LOWERSEQ;
6440
6441    REPEAT
6442
6443       CORRFILEFRAG (SEARCHLINES, HEAD, HIT, TAIL);
6444
6445       IF (HIT.FIRST <> NULLPOINTER)
6446          THEN
6447             BEGIN
6448
6449                SHIFTTEXT (UP, HEAD);
6450
6451                LINES := HIT;
6452                APPEND (LINES, HOLE);
6453
6454                LOWERSEQ := TAIL;
6455
6456                IF (REPLEN <> NULLSTRLEN)
6457                   THEN
6458                      PUTLINECHANGE;
6459
6460                SEARCHLINES := TAIL;
6461
6462                MATCHNUM := MATCHNUM - 1;
6463
```

```
6464              END
6465           ELSE
6466              IF (REQUEST.REPEATER <> REPEATALL) OR
6467                 (MATCHNUM = REQUEST.REPEATER)
6468                 THEN
6469                    BEGIN
6470                       SHOWERROR( ERRNOTFOUND ):
6471
6472                       IF (MATCHNUM <> REQUEST.REPEATER)
6473                          THEN
6474                             NOTENOUGH := TRUE:
6475                       END:
6476
6477     UNTIL DISABLE OR (MATCHNUM = 0) OR
6478           (HIT.FIRST = NULLPOINTER) OR ERROR:
6479
6480     IF NOT(DISABLE)
6481        THEN
6482           BEGIN
6483              GETTAIL (LOWERSEQ, TAILLINE, REMAININGLINES):
6484              SHIFTTEXT (UP, TAILLINE):
6485
6486              LOWERSEQ := REMAININGLINES:
6487
6488              GETLAST (UPPERSEQ, LEADINGLINES, LASTLINE):
6489
6490              UPPERSEQ := LEADINGLINES:
6491
6492              SHIFTTEXT (DOWN, LASTLINE):
6493              GETFIRST (LOWERSEQ, FIRSTLINE, REMAININGLINES):
6494
6495              CURRENTLINE := FIRSTLINE:
6496              LOWERSEQ    := REMAININGLINES:
6497
6498              IF ERROR
6499                 THEN
6500                    BEGIN
6501                       IF NOTENOUGH
6502                          THEN
6503                             SECURITYCHECK( DOREQUEST, ANSWER )
6504                          ELSE
6505                             ANSWER := NO:
6506
6507                       IF (ANSWER = NO)
6508                          THEN
6509                             TAKEBACKREQUEST:
6510                    END:
6511
6512              IF (ENDOFLINE)
6513                 THEN
6514                    BEGIN
6515                       WRITEFILE (CURRENTLINE):
6516                       SHOWLISTING:
6517                    END:
6518           END:
6519
6520  END: (* SEMMAKE *)
6521
6522  (*$7 *)
```

```
6523
6524 PROCEDURE SEMINSERT;
6525
6526 (*************************************************************************)
6527 (*                                                                       *)
6528 (* SEMINSERT PERFORMS THE SEMANTICS OF THE INSERT REQUEST. THE NEW       *)
6529 (* TEXT REQUESTED IS PLACED IN THE CORRECT POSITION RELATIVE TO THE      *)
6530 (* CURRENT LINE. IF THE NEW TEXT WAS ENTERED FROM THE TERMINAL OR        *)
6531 (* A FILE, A MESSAGE IS DISPLAYED. THE CURRENT LINE IS MOVED TO THE      *)
6532 (* LAST LINE OF THE NEW TEXT.                                            *)
6533 (*                                                                       *)
6534 (*************************************************************************)
6535
6536 VAR SEARCHLINES,
6537
6538     LASTLINE,
6539     LEADINGLINES,
6540
6541     FIRSTLINE,
6542     REMAININGLINES : INCORPTRS;
6543
6544     ANSWER : KEYSYMBOL;
6545
6546     I : INTEGER;
6547
6548 BEGIN (* SEMINSERT *)
6549
6550    LINESFOUND := 0;
6551
6552    CASE REQUEST.PLACE OF
6553
6554       AFTER  : BEGIN
6555                   SHIFTTEXT (UP, CURRENTLINE);
6556                   CURRENTLINE := NULLSEQ;
6557
6558                   GETNEWLINES;
6559
6560                   GETLAST (UPPERSEQ, LEADINGLINES, LASTLINE);
6561
6562                   UPPERSEQ := LEADINGLINES;
6563
6564                   SHIFTTEXT (DOWN, LASTLINE);
6565                   GETFIRST (LOWERSEQ, FIRSTLINE, REMAININGLINES);
6566
6567                   CURRENTLINE := FIRSTLINE;
6568                   LOWERSEQ    := REMAININGLINES;
6569                END;
6570
6571       BEFORE : BEGIN
6572                   APPEND (CURRENTLINE, LOWERSEQ);
6573
6574                   GETNEWLINES;
6575
6576                   GETLAST (UPPERSEQ, LEADINGLINES, LASTLINE);
6577
6578                   UPPERSEQ := LEADINGLINES;
6579
6580                   SHIFTTEXT (DOWN, LASTLINE);
6581                   GETFIRST (LOWERSEQ, FIRSTLINE, REMAININGLINES);
6582
6583                   CURRENTLINE := FIRSTLINE;
6584                   LOWERSEQ    := REMAININGLINES;
6585                END;
```

```
6586
6587        OVER    : BEGIN
6588                      APPEND (CURRENTLINE, HOLE):
6589
6590                      GETNEWLINES:
6591
6592                      GETLAST (UPPERSEQ, LEADINGLINES, LASTLINE):
6593
6594                      UPPERSEQ := LEADINGLINES:
6595
6596                      SHIFTTEXT (DOWN, LASTLINE):
6597                      GETFIRST (LOWERSEQ, FIRSTLINE, REMAININGLINES):
6598
6599                      CURRENTLINE := FIRSTLINE:
6600                      LOWERSEQ    := REMAININGLINES:
6601                  END:
6602
6603    END: (* CASE *)
6604
6605    IF NOT(DISABLE)
6606       THEN
6607          BEGIN
6608             IF ERROR
6609                THEN
6610                   TAKEBACKREQUEST:
6611
6612             IF (ENDOFLINE)
6613                THEN
6614                   BEGIN
6615                      WRITEFILE (CURRENTLINE):
6616                      SHOWLISTING:
6617                   END:
6618          END:
6619
6620 END: (* SEMINSERT *)
6621
6622 (*$Z *)
```

```
6623
6624 PROCEDURE SEMSHOW;
6625
6626 (*****************************************************************************)
6627 (*                                                                           *)
6628 (* SEMSHOW PERFORMS THE SEMANTICS OF THE SHOW REQUEST. THE INDIVIDUAL        *)
6629 (* ASSUMPTION IS SHOWN (OR ALL ASSUMPTIONS IF NONE WAS SPECIFIED).           *)
6630 (* IF AN INTERRUPT IS DETECTED, THE REMAINING ASSUMPTIONS WILL NOT           *)
6631 (* BE SHOWN.                                                                 *)
6632 (*                                                                           *)
6633 (*****************************************************************************)
6634
6635 VAR LINES : INCOREPTRS;
6636
6637     FIRSTLINE,
6638     REMAININGLINES,
6639
6640     LASTLINE,
6641     LEADINGLINES : INCOREPTRS;
6642
6643     FINISHED : BOOLEAN;
6644
6645     NAME,
6646     DATE : ALFA;
6647
6648     COUNT : INTEGER;
6649
6650 BEGIN (* SEMSHOW *)
6651
6652    IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6653       THEN
6654          WRITELN (MSGPREFIX);
6655
6656    IF (REQUEST.OPTION IN [MARGIN, ALL]) AND NOT(DISABLE)
6657       THEN
6658          SHOWVALUE (VALMARGINSYM);
6659
6660    IF (REQUEST.OPTION IN [JOKER, ALL]) AND NOT(DISABLE)
6661       THEN
6662          SHOWVALUE (VALJOKERSYM);
6663
6664    IF (REQUEST.OPTION IN [ELLIPSIS, ALL]) AND NOT(DISABLE)
6665       THEN
6666          SHOWVALUE (VALELLIPSIS);
6667
6668
6669    IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6670       THEN
6671          WRITELN (MSGPREFIX);
6672
6673    IF (REQUEST.OPTION IN [ALPHABET, ALL]) AND NOT(DISABLE)
6674       THEN
6675          SHOWVALUE (VALALPHABET);
6676
6677    IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6678       THEN
6679          WRITELN (MSGPREFIX);
6680
6681    IF (REQUEST.OPTION IN [FILESYM, ALL]) AND NOT(DISABLE)
6682       THEN
6683          SHOWVALUE (VALFILENAME);
6684
6685    IF (CURFILENAME = EMPTYFILE)
```

```
6686        THEN
6687           BEGIN
6688              IF (REQUEST.OPTION IN [TOP, BOTTOM]) AND NOT(DISABLE)
6689              THEN
6690                 SHOWERROR( ERRCANTSHOW ):
6691           END
6692        ELSE
6693           BEGIN
6694              IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6695              THEN
6696                 WRITELN (MSGPREFIX):
6697
6698              IF (REQUEST.OPTION IN [TOP, ALL]) AND NOT(DISABLE)
6699              THEN
6700                 BEGIN
6701                    SHOWVALUE (VALTOP):
6702
6703                    IF (UPPERSEQ.FIRST = NULLPOINTER)
6704                    THEN
6705                       FIRSTLINE := CURRENTLINE
6706                    ELSE
6707                       GETFIRST (UPPERSEQ,FIRSTLINE,REMAININGLINES):
6708
6709                    REWRITE (SCRATCH):
6710                    VALIDINFO := FALSE:
6711                    WRITEFILE (FIRSTLINE):
6712                    SHOWLISTING:
6713                 END:
6714
6715              IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6716              THEN
6717                 WRITELN (MSGPREFIX):
6718
6719              IF (REQUEST.OPTION IN [BOTTOM, ALL]) AND NOT(DISABLE)
6720              THEN
6721                 BEGIN
6722                    SHOWVALUE (VALBOTTOM):
6723
6724                    IF (LOWERSEQ.LAST = NULLPOINTER)
6725                    THEN
6726                       LASTLINE := CURRENTLINE
6727                    ELSE
6728                       GETLAST (LOWERSEQ, LEADINGLINES, LASTLINE):
6729
6730                    REWRITE (SCRATCH):
6731                    VALIDINFO := FALSE:
6732                    WRITEFILE (LASTLINE):
6733                    SHOWLISTING:
6734                 END:
6735           END:
6736
6737     IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6738        THEN
6739           WRITELN (MSGPREFIX):
6740
6741     IF (REQUEST.OPTION IN [CATALOG, ALL]) AND NOT(DISABLE)
6742        THEN
6743           BEGIN
6744              SHOWMESSAGE( CATLIST ):
6745              WRITELN (MSGPREFIX):
6746              SHOWMESSAGE( FILEHEAD ):
6747              WRITELN (MSGPREFIX):
6748
6749
```

```
6750                FINISHED := FALSE;
6751                COUNT := 0;
6752
6753                WHILE NOT(FINISHED) AND NOT(INTERRUPT) DO
6754                   BEGIN
6755                      NEXTFILE (NAME, DATE, FINISHED);
6756
6757                      IF (NAME [1] = COLON)
6758                         THEN
6759                            FINISHED := TRUE
6760                         ELSE
6761                            BEGIN
6762                               COUNT := COUNT + 1;
6763                               WRITE (MSGPREFIX);
6764                               WRITE (NAME, BLANK);
6765                               PRINTPACKED (DATE);
6766                               WRITELN;
6767                            END;
6768
6769                   END; (* WHILE *)
6770
6771                IF (COUNT = 0)
6772                   THEN
6773                      SHOWMESSAGE( EMPTYCAT )
6774                   ELSE
6775                      BEGIN
6776                         WRITELN (MSGPREFIX);
6777
6778                         WRITE (MSGPREFIX);
6779                         WRITE ('TOTAL :  ', COUNT : 1, BLANK);
6780                         PRINT (FILESYM);
6781                         WRITE ('(S)');
6782                         WRITELN;
6783                      END;
6784             END;
6785
6786       IF (REQUEST.OPTION = ALL) AND NOT(DISABLE)
6787          THEN
6788             WRITELN (MSGPREFIX);
6789
6790 END; (* SEMSHOW *)
6791
6792 (*$7 *)
```

```
6793
6794 PROCEDURE SEMPRESERVE:
6795
6796 (*************************************************************************)
6797 (*                                                                       *)
6798 (* SEMPRESERVE PERFORMS THE SEMANTICS OF THE PRESERVE REQUEST. IF THE *)
6799 (* CURRENT FILE WAS MODIFIED AND THE FILE WAS IN THE CATALOG (IE. THE *)
6800 (* FILE IS AN OLD FILE), A SECURITY CHECK IS MADE TO INSURE THAT THE  *)
6801 (* FILE SHOULD BE CHANGED.                                           *)
6802 (*                                                                   *)
6803 (*********************************************************************)
6804
6805 VAR ANSWER : KEYSYMBOL:
6806     SHOWNEW : BOOLEAN:
6807
6808 BEGIN (* SEMPRESERVE *)
6809
6810    SHOWNEW := FALSE:
6811
6812    IF (CURFILENAME = EMPTYFILE)
6813       THEN
6814          SHOWERROR( ERRNOCURFILE )
6815       ELSE
6816          IF (FILECHANGES > 0) OR
6817             (REQUEST.FILENAME <> CURFILENAME)
6818             THEN
6819                IF NOT(DISABLE)
6820                   THEN
6821                      BEGIN
6822                         IF HAVEFILE( REQUEST.FILENAME )
6823                            THEN
6824                               SECURITYCHECK( OVERWRITEFILE, ANSWER )
6825                            ELSE
6826                               BEGIN
6827                                SHOWNEW := TRUE:
6828                                ANSWER := YES:
6829                               END:
6830
6831                         IF (ANSWER = NO)
6832                            THEN
6833                               TAKEBACKREQUEST
6834                            ELSE
6835                               BEGIN
6836                                WRITEFILE (TOPSEQ):
6837                                WRITEFILE (UPPERSEQ):
6838                                WRITEFILE (CURRENTLINE):
6839                                WRITEFILE (LOWERSEQ):
6840                                WRITEFILE (BOTTOMSEQ):
6841
6842                                PUTINTO (REQUEST.FILENAME):
6843
6844                                IF (SHOWNEW)
6845                                 THEN
6846                                  SHOWMESSAGE( NEWFILE ):
6847
6848                                IF (REQUEST.FILENAME = CURFILENAME)
6849                                 THEN
6850                                  BEGIN
6851                                   FILECHANGES := 0:
6852                                   FILESTATUS := OLD:
6853                                  END:
6854                               END:
6855                      END:
```

```
6856
6857 END: (* SEMPRESERVE *)
6858
6859 (*$Z *)


6860
6861 PROCEDURE SEMDESTROY;
6862
6863 (*************************************************************************)
6864 (*                                                                       *)
6865 (* SEMDESTROY PERFORMS THE SEMANTICS OF THE DESTROY REQUEST. IF THE      *)
6866 (* FILE IS FOUND IN THE CATALOG (IE. THE FILE IS AN OLD FILE), A         *)
6867 (* SECURITY CHECK IS MADE TO INSURE THAT THE FILE SHOULD BE REMOVED.     *)
6868 (*                                                                       *)
6869 (*************************************************************************)
6870
6871 VAR ANSWER : KEYSYMBOL;
6872
6873 BEGIN (* SEMDESTROY *)
6874
6875    IF (REQUEST.FILENAME = EMPTYFILE)
6876       THEN
6877          SHOWERROR( ERRNOCURFILE )
6878       ELSE
6879          IF NOT(DISABLE)
6880             THEN
6881                BEGIN
6882                   IF HAVEFILE( REQUEST.FILENAME )
6883                      THEN
6884                         SECURITYCHECK( ELIMINATEFILE, ANSWER )
6885                      ELSE
6886                         BEGIN
6887                            SHOWMESSAGE( NOFILEFOUND );
6888                            ANSWER := NO;
6889                         END;
6890
6891                   IF (ANSWER = NO)
6892                      THEN
6893                         TAKEBACKREQUEST
6894                      ELSE
6895                         BEGIN
6896                            REMOVE (REQUEST.FILENAME);
6897
6898                            IF (REQUEST.FILENAME = CURFILENAME)
6899                               THEN
6900                                  FILESTATUS := NEW;
6901                            END;
6902                END;
6903
6904 END; (* SEMDESTROY *)
6905
6906 (*$Z *)
```

```
6907
6908 PROCEDURE SEMASSUME:
6909
6910 (*************************************************************************)
6911 (*                                                                       *)
6912 (* SEMASSUME PERFORMS THE SEMANTICS OF THE ASSUME REQUEST. IF THE        *)
6913 (* CURRENT FILE WAS MODIFED OR THE FILE IS NOT IN THE CATALOG (IE.       *)
6914 (* THE FILE IS A NEW FILE), A CHECK IS MADE BEFORE A NEW FILE IS         *)
6915 (* ASSUMED IN ORDER TO PROTECT AGAINST LOSING THE CURRENT FILE.          *)
6916 (*                                                                       *)
6917 (*************************************************************************)
6918
6919 VAR SEARCHLINES,
6920     FIRSTLINE,
6921     REMAININGLINES : INCOREPTRS:
6922
6923     ANSWER : KEYSYMBOL:
6924
6925 BEGIN (* SEMASSUME *)
6926
6927    IF (REQUEST.OPTION IN [TOP, BOTTOM]) AND
6928       (CURFILENAME = EMPTYFILE)
6929       THEN
6930          SHOWERROR( ERRNOCURFILE )
6931       ELSE
6932          CASE REQUEST.OPTION OF
6933
6934             TOP       : IF (REQUEST.VALUE = HERE)
6935                            THEN
6936                               APPEND (UPPERSEQ, TOPSEQ)
6937                            ELSE
6938                               APPEND (TOPSEQ, UPPERSEQ):
6939
6940             BOTTOM    : IF (REQUEST.VALUE = HERE)
6941                            THEN
6942                               APPEND (LOWERSEQ, BOTTOMSEQ)
6943                            ELSE
6944                               APPEND (BOTTOMSEQ, LOWERSEQ):
6945
6946             MARGIN    : MARGINSYM    := REQUEST.SPECIALSYM:
6947
6948             JOKER     : JOKERSYM     := REQUEST.SPECIALSYM:
6949
6950             ELLIPSIS  : ELLIPSISSYM  := REQUEST.SPECIALSYM:
6951
6952             ALPHABET  : BEGIN
6953                            CURALPHABET := REQUEST.VALUE:
6954
6955                            IF (CURALPHABET = ASCII)
6956                               THEN
6957                                  SETASCII
6958                               ELSE
6959                                  SETNORMAL:
6960                         END:
6961
6962             FILESYM   : BEGIN
6963
6964                            LINESFOUND := 0:
6965
6966                            IF (FILECHANGES > 0)
6967                               THEN
6968                                  SECURITYCHECK( FORGETFILE, ANSWER )
6969                               ELSE
```

```
6970                                    ANSWER := YES:
6971
6972                                IF (ANSWER = NO)
6973                                   THEN
6974                                      TAKEBACKREQUEST
6975                                   ELSE
6976                                      BEGIN
6977
6978                                       CURFILENAME := REQUEST.FILENAME:
6979                                       SETUP (CURFILENAME):
6980
6981                                       TOPSEQ      := NULLSEQ:
6982                                       UPPERSEQ    := NULLSEQ:
6983                                       HOLE        := MAXSEQ:
6984                                       CURRENTLINE := NULLSEQ:
6985                                       LOWERSEQ    := NULLSEQ:
6986                                       BOTTOMSEQ   := NULLSEQ:
6987
6988                                       FILECHANGES := 0:
6989
6990
6991                                       READFILE:
6992
6993                                       IF ERROR
6994                                        THEN
6995                                         TAKEBACKREQUEST
6996                                        ELSE
6997                                         BEGIN
6998
6999                                          SHIFTTEXT (DOWN, UPPERSEQ):
7000
7001                                          GETFIRST (LOWERSEQ, FIRSTLINE,
7002                                                    REMAININGLINES):
7003
7004                                          TOPSEQ      := NULLSEQ:
7005                                          UPPERSEQ    := NULLSEQ:
7006                                          CURRENTLINE := FIRSTLINE:
7007                                          LOWERSEQ    := REMAININGLINES:
7008                                          BOTTOMSEQ   := NULLSEQ:
7009
7010                                          REWRITE (SCRATCH):
7011                                          VALIDINFO := FALSE:
7012                                          SHOWVALUE( VALCURLINE ):
7013
7014                                         END:
7015                                      END:
7016
7017                                WRITEFILE (CURRENTLINE):
7018                                SHOWLISTING:
7019
7020                             END:
7021
7022          END: (* CASE *)
7023
7024 END: (* SEMASSUME *)
7025
7026 (*$Z *)
```

```
7027
7028 PROCEDURE SEMGRIPE;
7029
7030 (*****************************************************************)
7031 (*                                                               *)
7032 (* SEMGRIPE PERFORMS THE SEMANTICS OF THE GRIPE REQUEST. THE USER *)
7033 (* NUMBER, DATE AND TIME ARE SAVED. THE USER IS THEN PROMPTED TO  *)
7034 (* THE ENTER GRIPE (UNTIL INTERRUPTED).                          *)
7035 (*                                                               *)
7036 (*****************************************************************)
7037
7038 VAR I : STRINGINDEX;
7039
7040     LINE : STRING;
7041
7042     LENGTH : STRINGINDEX;
7043
7044     FINISHED : BOOLEAN;
7045
7046     A, B, C : ALFA;
7047
7048 BEGIN (* SEMGRIPE *)
7049
7050    VALIDINFO := FALSE;
7051    FINISHED := FALSE;
7052
7053    IF NOT(DISABLE)
7054       THEN
7055          BEGIN
7056             SHOWMESSAGE( ENTERGRIPE );
7057
7058             GETUSER (A);
7059             DATE (B);
7060             TIME (C);
7061             WRITELN (COMMENT);
7062             WRITELN (COMMENT, A, B, C);
7063             WRITELN (COMMENT);
7064
7065             REPEAT
7066
7067                GETLINE( CHKPROMPT, LINE, LENGTH );
7068
7069                IF INTERRUPT
7070                   THEN
7071                      FINISHED := TRUE
7072                   ELSE
7073                      BEGIN
7074                         IF (LENGTH <> NULLSTRLEN)
7075                            THEN
7076                               FOR I := 1 TO LENGTH DO
7077                                WRITE (COMMENT, LINE[I]);
7078
7079                         WRITELN (COMMENT);
7080                      END;
7081
7082             UNTIL FINISHED;
7083
7084             WRITELN (COMMENT);
7085
7086             REMOVE (REQUEST.FILENAME);
7087
7088             IF (REQUEST.FILENAME = CURFILENAME)
7089                THEN
```

```
7090                    FILESTATUS := NEW;
7091
7092                SHOWMESSAGE( GRIPESAVED );
7093            END;
7094
7095 END; (* SEMGRIPE *)
7096
7097 (*!7 *)
```

```
7098
7099 PROCEDURE SEMUNDO;
7100
7101 (*************************************************************************)
7102 (*                                                                       *)
7103 (* SEMUNDO PERFORMS THE SEMANTICS OF THE UNDO REQUEST. IF THE REQUEST    *)
7104 (* CAN BE UNDONE, THE STATE IS RESET. IF A CURRENT LINE WAS DISPLAYED    *)
7105 (* BEFORE THE REQUEST WAS PERFORMED, THE CURRENT LINE IS DISPLAYED       *)
7106 (* AGAIN.                                                                *)
7107 (*                                                                       *)
7108 (*************************************************************************)
7109
7110 BEGIN (* SEMUNDO *)
7111
7112    IF LASTREQUEST.NAME IN [SHOW..UNDO, GRIPE, UNKNOWN]
7113       THEN
7114          SHOWERROR( ERRUNDO )
7115       ELSE
7116          IF NOT(DISABLE)
7117             THEN
7118                BEGIN
7119
7120                   TOPSEQ       := TEMPTOPSEQ;
7121                   UPPERSEQ     := TEMPUPPERSEQ;
7122                   HOLE         := TEMPHOLE;
7123                   CURRENTLINE  := TEMPCURRENTLINE;
7124                   LOWERSEQ     := TEMPLOWERSEQ;
7125                   BOTTOMSEQ    := TEMPBOTTOMSEQ;
7126
7127                   INCOPYTEXT   := TEMPINCOPYTEXT;
7128
7129
7130                   FILECHANGES  := TEMPFILECHANGES;
7131                   FILESTATUS   := TEMPFILESTATUS;
7132
7133                   CURFILENAME  := TEMPCURFILENAME;
7134
7135                   CURALPHABET  := TEMPCURALPHABET;
7136
7137                   MARGINSYM    := TEMPMARGINSYM;
7138                   JOKERSYM     := TEMPJOKERSYM;
7139                   ELLIPSISSYM  := TEMPELLIPSISSYM;
7140
7141                   IF LASTREQUEST.NAME IN [TRANSFER, COPY]
7142                      THEN
7143                         SHOWMESSAGE( CANTUNDO );
7144
7145                   SHOWMESSAGE( UNDONE );
7146
7147                   IF (LASTREQUEST.NAME IN [FORWARD..INSERT]) OR
7148                      (CURFILENAME <> EMPTYFILE) AND
7149                      (LASTREQUEST.OPTION IN [FILESYM, TOP, BOTTOM])
7150                      THEN
7151                         BEGIN
7152                            SHOWVALUE( VALCURLINE );
7153                            WRITEFILE (CURRENTLINE);
7154                            SHOWLISTING;
7155                         END;
7156
7157                END;
7158
7159 END; (* SEMUNDO *)
7160
```

```
7161 (*$7 *)
7162
7163 PROCEDURE SEMREQUEST;
7164
7165 (*****************************************************************)
7166 (*                                                               *)
7167 (* SEMREQUEST PERFORMS THE SEMANTICS OF THE CURRENT REQUEST. IF THE   *)
7168 (* REQUEST WOULD MODIFY THE STATE OF THE EDITOR, THE STATE IS SAVED   *)
7169 (* PRIOR TO ITS EXECUTION IN ORDER TO FACILITATE AN UNDO REQUEST.     *)
7170 (*                                                               *)
7171 (* IF AN INTERRUPT IS DETECTED AFTER COMPLETION OF A REQUEST, THE     *)
7172 (* AFFECTS OF THAT REQUEST WILL BE UNDONE.                           *)
7173 (*                                                               *)
7174 (*****************************************************************)
7175
7176 BEGIN (* SEMREQUEST *)
7177
7178    REWRITE (SCRATCH);
7179    VALIDINFO := FALSE;
7180
7181    IF REQUEST.NAME IN [SHOW..UNDO, GRIPE]
7182       THEN
7183          BEGIN
7184
7185             CASE REQUEST.NAME OF
7186
7187                SHOW     : SEMSHOW;
7188
7189                PRESERVE : SEMPRESERVE;
7190
7191                DESTROY  : SEMDESTROY;
7192
7193                UNDO     : SEMUNDO;
7194
7195                GRIPE    : SEMGRIPE;
7196
7197             END; (* CASE *)
7198
7199             IF (INTERRUPT) AND NOT(ERROR)
7200                THEN
7201                   TAKEBACKREQUEST;
7202
7203          END
7204       ELSE
7205          IF (CURFILENAME = EMPTYFILE) AND
7206             (REQUEST.NAME <> ASSUME)
7207             THEN
7208                SHOWERROR( ERRNOCURFILE )
7209             ELSE
7210                BEGIN
7211
7212                   TEMPTOPSEQ       := TOPSEQ;
7213                   TEMPUPPERSEQ     := UPPERSEQ;
7214                   TEMPHOLE         := HOLE;
7215                   TEMPCURRENTLINE  := CURRENTLINE;
7216                   TEMPLOWERSEQ     := LOWERSEQ;
7217                   TEMPBOTTOMSEQ    := BOTTOMSEQ;
7218
7219                   TEMPINCOPETEXT   := INCOPETEXT;
7220
7221
7222                   TEMPFILECHANGES  := FILECHANGES;
7223                   TEMPFILESTATUS   := FILESTATUS;
7224
```

```
7225                    TEMPCURFILENAME := CURFILENAME:
7226
7227                    TEMPCURALPHABET := CURALPHABET:
7228
7229                    TEMPMARGINSYM   := MARGINSYM:
7230                    TEMPELLIPSISSYM := ELLIPSISSYM:
7231                    TEMPJOKERSYM    := JOKERSYM:
7232
7233                    IF REQUEST.NAME IN [DELETE, TRANSFER, MAKE, INSERT]
7234                       THEN
7235                          FILECHANGES := FILECHANGES + 1:
7236
7237                    CASE REQUEST.NAME OF
7238
7239                       FORWARD  : SEMFORWARD:
7240
7241                       BACKWARD : SEMBACKWARD:
7242
7243                       LIST     : SEMLIST:
7244
7245                       DELETE   : SEMDELETE:
7246
7247                       TRANSFER : SEMTRANSFER:
7248
7249                       COPY     : SEMCOPY:
7250
7251                       MAKE     : SEMMAKE:
7252
7253                       INSERT   : SEMINSERT:
7254
7255                       ASSUME   : SEMASSUME:
7256
7257                    END: (* CASE *)
7258
7259                    IF (INTERRUPT) AND NOT(ERROR)
7260                       THEN
7261                          BEGIN
7262                             TAKEBACKREQUEST:
7263
7264                             IF (CURFILENAME <> EMPTYFILE)
7265                                THEN
7266                                 BEGIN
7267                                  REWRITE (SCRATCH):
7268                                  VALIDINFO := FALSE:
7269                                  WRITEFILE (CURRENTLINE):
7270                                  SHOWLISTING:
7271                                 END:
7272                          END:
7273
7274                 END:
7275
7276 END: (* SEMREQUEST *)
7277
7278 (*$Z *)
```

```
7279
7280 BEGIN (* HOPE *)
7281
7282    PROTECT;
7283
7284    SHOWMESSAGE( INTRO );
7285
7286    INITIALIZE;
7287
7288    REPEAT
7289
7290       PARSEREQUEST;
7291
7292       IF NOT(INTERRUPT)
7293          THEN
7294             IF (REQUEST.NAME <> QUIT)
7295                THEN
7296                   SEMREQUEST
7297                ELSE
7298                   BEGIN
7299
7300                      IF (FILECHANGES > 0)
7301                         THEN
7302                            SECURITYCHECK( OKTOQUIT, ANSWER )
7303                         ELSE
7304                            ANSWER := YES;
7305
7306                      IF (ANSWER = NO)
7307                         THEN
7308                            TAKEBACKREQUEST
7309                         ELSE
7310                            DONE := TRUE;
7311
7312                   END;
7313
7314       IF ERROR
7315          THEN
7316             LASTREQUEST := NULLREQUEST
7317          ELSE
7318             LASTREQUEST := REQUEST;
7319
7320    UNTIL DONE;
7321
7322    SHOWMESSAGE( FINAL );
7323
7324    SETBACK;
7325
7326 END. (* HOPE *)
```

UMASS DEPENDANT COMPASS SUBROUTINES SOURCE LISTING

```
  1                 IDENT HOPESYS      HOPE CDC SYSTEM ROUTINES
  2 *
  3                 ENTRY PROTECT
  4                 ENTRY INTERRU
  5                 ENTRY DISABLE
  6                 ENTRY GETUSER
  7                 ENTRY GETASCT
  8                 ENTRY SETNORM
  9                 ENTRY SETBACK
 10                 ENTRY HAVEFIL
 11                 ENTRY NEXTFIL
 12                 ENTRY FIND
 13                 ENTRY SAVE
 14                 ENTRY REMOVE
 15  NOSCOMS        XTEXT COMCMAC
 16  NOSCOMS        XTEXT COMSPFM
 17                 EXT   P.OPEN,P.CLOSE
 18 *
 19 *
 20  INTADDR        BSSZ  1
 21  PFN            BSSZ  1
 22  LFN            BSSZ  1
 23  CAT            FILEB BUFF,80,(FET=9)
 24  BUFF           BSSZ  80
 25  SAVEB2         BSSZ  1
 26  LFNFET         FILEB BUFF1,1,(FET=9)
 27  BUFF1          BSSZ  1
 28  OKADDR         BSSZ  1
 29  EFET           BSSZ  1
 30  BUFADDR        BSSZ  1
 31  BUFLEN         BSSZ  1
 32  BLANK          DATA  55P
 33  BLANKS         DIS   ,'                '
 34  COLON          DATA  00P
 35  UNUM           BSSZ  1
 36  ADDR           BSSZ  1
 37  NAMADDR        BSSZ  1
 38  DATADDR        BSSZ  1
 39  FINADDR        BSSZ  1
 40  TRUE           DATA  1
 41  FALSE          DATA  0
 42 **
 43 *
 44 *
 45  PROTECT        BSS   1
 46                 EREXIT 0
 47                 DISTC ON,INTADDR   TURN ON INTERRUPT CAPABILITY
 48                 MX6   0
 49                 SA6   INTADDR      CLEAR INTERRUPT COUNTER
 50                 EQ    PROTECT
 51 **
 52 *
 53 *
 54  INTERRU        BSS   1
 55                 SA1   INTADDR      CHECK LAST INTERRUPT STATUS
 56                 MX6   0
 57                 SA6   INTADDR      CLEAR ANY INTERRUPTS
 58                 NZ    X1,I1
 59                 SA5   FALSE        NO INTERRUPTS DETECTED
 60                 BX6   X5
 61                 EQ    I2
```

```
 62  I1          SA5    TRUE          INTERRUPT SIGNALLED
 63              BX6    X5
 64  I2          EQ     INTERRU
 65 **
 66 *
 67 *
 68  DISABLE     BSS    1
 69              SA1    INTADDR       CHECK LAST INTERRUPT STATUS
 70              NZ     X1,D1
 71              SA5    FALSE         NO INTERRUPTS DETECTED
 72              BX6    X5
 73              EQ     D2
 74  D1          SA5    TRUE          INTERRUPT SIGNALLED
 75              BX6    X5
 76  D2          EQ     DISABLE
 77 **
 78 *
 79 *
 80  GETUSER     BSS    1
 81              SA1    A1
 82              BX6    X1
 83              SA6    ADDR
 84              USERNUM UNUM         GET THE USER NUMBER
 85              SA5    UNUM
 86              SA2    BLANKS
 87              MX0    42
 88              BX5    X0*X5
 89              BX2    -X0*X2
 90              BX6    X5+X2         CHANGE ZERO BYTES TO BLANK
 91              SA1    ADDR
 92              SA6    X1
 93              EQ     GETUSER
 94 *
 95 *
 96 *
 97  SETASCI     BSS    1
 98              CSET   ASCII         SET TERMINAL TO ASCII MODE
 99              EQ     SETASCI
100 *
101 *
102 *
103  SETBACK     BSS    1
104              CSET   RESTORE       RETURN TO ORIGINAL MODE
105              EQ     SETBACK
106 *
107 *
108 *
109  SETNORM     BSS    1
110              CSET   NORMAL        SET TERMINAL TO NORMAL MODE
111              EQ     SETNORM
112 **
113 *
114 *
115  ZERO        BSS    1             CHANGE BLANKS TO ZERO BYTES
116              MX6    0
117              SX6    B2
118              SA6    SAVEB2        SAVE THE CONTENTS OF B2
119              SA2    X1
120              BX6    X2            GET FILENAME
121              MX3    6
122              SA2    BLANK
123              MX0    6
124              LX0    -6
125              SB2    48
```

```
126 *
127  LOOP       BX4    X0*X6          FIND ALL BLANKS
128             LX4    -B2
129             IX4    X4-X2
130             ZR     X4,DONE
131 *
132             SB2    B2-6
133             AX3    6              MASK FOR FILENAME
134             LX0    -6
135             ZR     B2,DONE
136             EQ     LOOP
137 *
138  DONE       BX6    X6*X3          MASK OUT BLANKS
139             SA5    SAVEB2
140             SB2    X5
141             EQ     ZERO
142 **
143 *
144 *
145  HAVEFIL    BSS    1              FIND FILE IN CATALOG
146             RJ     ZERO
147 *
148             SA6    PFN            SAVE THE FILENAME
149             SA6    CAT+8
150 *
151             MX0    42             CLEAR THE FET
152             SA2    CAT
153             BX6    X0*X2
154             SX0    3
155             BX6    X0+X6
156             SA6    CAT
157 *
158             MX0    1              REQUEST ERROR CONTROL
159             LX0    45
160             SA1    CAT+1
161             BX6    X1+X0
162             SA6    CAT+1
163 *
164             CATLIST CAT,PFN       CHECK FOR FILE
165             MX0    4
166             LX0    14
167             SA1    CAT
168             BX1    X1*X0
169             MX0    -0
170             SA5    CAT+5
171             BX7    -X0*X5
172             SA7    CAT+5
173             NZ     X1,H1
174 *
175             SA5    TRUE           EVERYTHING IS OK
176             BX6    X5
177             EQ     H2
178  H1         SA5    FALSE          NO FILE FOUND
179             BX6    X5
180  H2         EQ     HAVEFIL
181 **
182 *
183 *
184  CLEAR      BSS    1              CHANGE ZERO BYTES TO BLANKS
185             MX7    0
186             SX7    B2
187             SA7    SAVEB2         SAVE THE CONTENTS OF B2
188             MX3    6
189             SA2    COLON
```

```
190             MX0     6
191             LX0     -6
192             SB2     48
193 *
194  BL1        BX4     X0*X6           FIND ALL ZERO BYTES
195             LX4     -B2
196             IX4     X4-X2
197             ZR      X4,BL2
198 *
199             SB2     B2-6
200             AX3     6               MASK FOR NAME
201             LX0     -6
202             ZR      B2,BL2
203             EQ      BL1
204 *
205  BL2        SA5     BLANKS          MASK OUT ZERO BYTES
206             BX5     -X3*X5
207             BX6     X6+X5
208             SA5     SAVEB2
209             SB2     X5
210             EQ      CLEAR
211 **
212 *
213 *
214  NEXTFIL    BSS     1               RETURN NEXT FILE INFORMATION
215             MX0     0
216             BX6     X0
217             SA6     CAT+6
218             SA2     A1
219             BX6     X2
220             SA6     NAMADDR         SAVE NAME ADDRESS
221             SA1     A1+1
222             SA2     A1
223             BX6     X2
224             SA6     DATADDR         SAVE DATE ADDRESS
225             SA1     A1+1
226             SA2     A1
227             BX6     X2
228             SA6     FINADDR         SAVE FINISH ADDRESS
229             SA1     CAT+3
230             SA2     CAT+2
231             BX2     X2-X1
232             ZR      X2,GETMORE      IF EMPTY, ASK FOR MORE
233 *
234  EXIT       SA5     X1
235             MX0     42
236             BX6     X5*X0
237             RJ      CLEAR
238 *
239             SA5     X1+3
240             BX7     X5
241             SA5     X1+5
242             SA1     NAMADDR         RETURN THE PARAMETERS
243             SA6     X1
244             SA1     DATADDR
245             SA7     X1
246             SA1     FINADDR
247             BX6     X5
248             SA6     X1
249             SA1     CAT+3
250             SX7     X1+8
251             SA7     CAT+3           UPDATE FET POINTERS
252             EQ      NEXTFIL
253 *
```

```
254  GETMORE   SA1   CAT+1
255            SX6   X1
256            SA6   CAT+2          RESET THE BUFFER
257            SA6   CAT+3
258            CATLIST CAT
259            SA1   CAT+3          POINTER TO ENTRY
260 *
261            SA2   CAT
262            MX0   -10
263            BX2   -X0*X2
264            SX2   X2-338
265            JR    X2,EXIT        IF NOT EMPTY
266 *
267            SA2   CAT+2
268            SA5   TRUE
269            SX6   X5
270            SA3   CAT+1
271            IX3   X3-X2
272            JR    X3,G1
273 *
274            SA6   X2-3
275  G1        MX0   30
276            SA5   CAT+6
277            BX6   -X0*X5
278            SA6   CAT+6
279            EQ    EXIT
280 **
281 *
282 *
283  FIND      BSS   1              GET THE FILE FROM CATALOG
284            RJ    ZERO
285 *
286            SA6   PFN
287            SA2   A1+2           GET ADDRESS OF OK PARAMETER
288            BX6   X2
289            SA6   OKADDR
290            SA1   A1+1           ADDRESS OF LFN EFET-13
291 *
292            SX1   X1+13          GET EFET
293            SX6   X1
294            SA6   EFET
295            SA2   X1+1
296            MX0   42
297            BX6   X0*X2          GET FILENAME
298            SA6   LFN
299 *
300            SA6   LFNFET+8
301            SX0   3
302            BX6   X6+X0
303            SA6   LFNFET
304            SA2   A2+1
305            SX6   X2
306            SA6   BUFADDR        STORE FIRST
307            SA3   A2+3
308            SX6   X3
309            IX6   X6-X2          FIRST - LAST
310            SA6   BUFLEN
311            SA1   X1
312            RJ    P.CLOSE        CLOSE THE CURRENT FILE
313 *
314            MX0   1              SET ERROR CONTROL
315            LX0   45
316            SA1   LFNFET+1
317            BX1   X1+X0
```

```
318             BX6     X1
319             SA6     LFNFET+1
320             GET     LFNFET,PFN
321             MX0     8
322             LX0     18
323             SA1     LFNFET
324             BX1     X1*X0
325             NZ      X1,F1           IF FILE NOT FOUND
326 *
327             SA1     EFET            SET UP NEW FILE
328             SA0     X1
329             SX1     23
330             SA2     LFN
331             SA3     BUFLEN
332             SX6     B1
333             SA4     BUFADDR
334             SX7     X4
335             RJ      P.OPEN          OPEN THE NEW FILE
336 *
337             SX6     TRUE            EVERYTHING IS OK
338             SA2     OKADDR
339             SA6     X2
340             EQ      FIND
341 *
342  F1         SA5     FALSE           SOMETHING IS WRONG
343             BX6     X5
344             SA2     OKADDR
345             SA6     X2
346             EQ      FIND
347 **
348 *
349 *
350  SAVE       BSS     1
351             RJ      ZERO
352 *
353             SA6     PFN
354             SA2     A1+2            GET ADDRESS OF OK PARAMETER
355             BX6     X2
356             SA6     OKADDR
357             SA1     A1+1            ADDRESS OF LFN EFET-13
358 *
359             SX1     X1+13           GET EFET
360             SX6     X1
361             SA6     EFET
362             SA2     X1+B1
363             MX0     42
364             BX6     X0*X2           GET FILENAME
365             SA6     LFN
366 *
367             SA6     LFNFET+8
368             SX0     3
369             BX6     X6+X0
370             SA6     LFNFET
371             SA2     A2+1
372             SX6     X2
373             SA6     BUFADDR         STORE FIRST
374             SA3     A2+3
375             SX6     X3
376             IX6     X6-X2           FIRST - LAST
377             SA6     BUFLEN
378             SA1     X1
379             RJ      P.CLOSE         CLOSE THE CURRENT FILE
380 *
381             MX0     1               SET ERROR CONTROL
```

```
382                 LX0   45
383                 SA1   LFNFET+1
384                 BX1   X1+X0
385                 BX6   X1
386                 SA6   LFNFET+1
387                 REPLACE LFNFET,PFN
388                 MX0   8
389                 LX0   18
390                 SA1   LFNFET
391                 BX1   X1*X0
392                 NZ    X1,S1         IF CAN'T SAVE FILE
393 *
394                 SA1   EFET          SET UP NEW FILE
395                 SA0   X1
396                 SX1   23
397                 SA2   LFN
398                 SA3   BUFLEN
399                 SX6   B1
400                 SA4   BUFADDR
401                 SX7   X4
402                 RJ    P.OPEN        OPEN THE NEW FILE
403 *
404                 SA5   TRUE          EVERYTHING IS OK
405                 BX6   X5
406                 SA2   OKADDR
407                 SA6   X2
408                 EQ    SAVE
409 *
410  S1             SA5   FALSE         SOMETHING IS WRONG
411                 BX6   X5
412                 SA2   OKADDR
413                 SA6   X2
414                 EQ    SAVE
415 **
416 *
417 *
418  REMOVE         BSS   1             PURGE THE FILE
419                 RJ    ZERO
420 *
421                 SA6   LFNFET+8
422                 SX0   3
423                 BX6   X6+X0
424                 SA6   LFNFET
425 *
426                 MX0   1             SET ERROR CONTROL
427                 LX0   45
428                 SA1   LFNFET+1
429                 BX1   X1+X0
430                 BX6   X1
431                 SA6   LFNFET+1
432 *
433                 PURGE LFNFET
434                 EQ    REMOVE
435 *
436 *
437                 END
```

	918	920	925	927	932	934	938	940	942	944
	948	950	954	956	958	960	962	966	968	970
	972	974	978	980	982	984	986	990	992	994
	998	1000	1005	1011	1017	1020	1027	1028	1032	1033
	1037	1039	1043	1045	1049	1051	1055	1057	1062	1063
	1067	1069	1071	1093	1098	1100	1102	1111	1116	1118
	1120	1121	1127	1130	1131	1135	1138	1139	1142	1147
	1152	1154	1155	1162	1165	1166	1167	1169	1170	1175
	1177	1182	1185	1186	1190	1192	1194	1198	1200	1204
	1205	1209	1212	1214	1215	1218	1220	1221	1222	1226
	1228	1250	1257	1259	1260	1264	1269	1271	1274	1281
	1284	1291	1294	1301	1307	1309	1316	1322	1327	1329
	1336	1369	1374	1376	1381	1390	1393	1397	1398	1399
	1402	1403	1405	1410	1411	1416	1418	1420	1421	1426
	1430	1433	1435	1439	1441	1445	1447	1450	1451	1452
	1456	1458	1462	1465	1468	1470	1473	1475	1479	1482
	1484	1487	1491	1493	1498	1500	1504	1509	1514	1516
	1594	1599	2072	2076	2085	2086	2090	2096	2097	2101
	2104	2107	2110	2155	2185	2191	2196	2240	2242	2243
	2244	2246	2247	2251	2252	2257	2269	2270	2271	2273
	2274	2278	2279	4275	4283	4285	4289	4292	4294	4299
	4301	4305	4308	4310	4314	4317	4319	4320	4321	4322
	4326	4327	4328	4331	4333	4334	4335	4336	4348	4354
	4355	4363	4364	4674	5384	5428	6767	6764	6778	5779
	6781	7077								
YES	115	555	1440	2014	2016	2033	2062	2147	2180	2226
	4373	4593	4613	4618	4623	6240	6367	6828	6979	7304
ZERO	787	810	830							

459 IDENTIFIERS 6680 OCCURRENCES