```
*                                                                           *)
* ADVENTURE IN PASCAL - MARCH 1979                                          *)
*                                                                           *)
* WRITTEN BY                                                                *)
*    GEORGE H. RICHMOND                                                     *)
*    STORAGE TECHNOLOGY CORPORATION                                         *)
*                                                                           *)
* WITH THE SUPPORT OF                                                       *)
*    MIKE PRESTON                                                           *)
*    UNIVERSITY OF COLORADO AT BOULDER                                      *)
*    ENGLISH DEPARTMENT                                                     *)
*                                                                           *)
* AND THE                                                                   *)
*    UNIVERSITY OF COLORADO COMPUTING CENTER                                *)
*                                                                           *)
*  DATA FILE FORMAT.                                                        *)
*                                                                           *)
*  1) MESSAGE OF THE DAY.                                                   *)
*  2) DIRECTIONS, ACTIONS, AND NOUNS.                                       *)
*  3) NOUN DESCRIPTIONS. (+ PREFIX)                                         *)
*       - EITHER 1 LINE SHORT + N LINES LONG DESCRIPTIONS,                  *)
*       - OR     1 LINE PER STATE OF NOUN.                                  *)
*  4) SYNONYMS. (- PREFIX)                                                  *)
*  5) ROOM DESCRIPTIONS. (+ PREFIX)                                         *)
*       - LINE  1    COLUMN 2   BLANK = LAND                                *)
*                               B     = SHORE OR BEACH                      *)
*                               W     = LAKE                                *)
*                    COLUMN 3   BLANK = NO POOLS                            *)
*                               O     = POOL OF OIL                         *)
*                               W     = POOL OF WATER                       *)
*                    COLUMN 4   BLANK = INSIDE CAVERN                       *)
*                               O     = OUTSIDE CAVERN                      *)
*                    COLUMNS 5-70     = INTERNAL NAME                       *)
*       - LINE  2          BRIEF DESCRIPTION                                *)
*       - LINES 3-25       LONG DESCRIPTION                                 *)
*  6) INTERCONNECTIONS. (- PREFIX)                                          *)
*       A) FROM,                                                            *)
*       B) TO,                                                              *)
*       C) DIRECTION,                                                       *)
*       D) OPTIONAL BLOCK NUMBER.                                           *)
*          (REPEAT C,D AS NECESSARY)                                        *)
*  7) OBJECT PLACEMENT. (+ PREFIX)                                          *)
*       A) LOCATION,                                                        *)
*       B) LIST OF NOUNS.                                                   *)
*  8) SPECIAL LOCATIONS (- PREFIX)                                          *)
*       A) STARTING AND RESURRECTION POSITION,                              *)
*       B) TREASURE DEPOSIT POSITION,                                       *)
*       C) BOAT STARTING AND RESURRECTION POSITION.                         *)

ROGRAM ADVENTURE ( INPUT/+, OUTPUT, ADVENT4 );
*$E-*)

ABEL 50; (* GO ASK QUESTION *)
*$N TYPE DEFINITIONS                                                        *)

ONST WORDSIZE     = 20;  (* NUMBER OF CHARACTERS IN A STRING *)
```

```
 LINEWIDTH      = 70;  (* NUMBER OF CHARACTERS IN A LINE   *)
 DESCMAX        = 10;  (* MAXIMUM LINES IN A DESCRIPTION   *)
 NOBLOCKS       = 20;  (* MAXIMUM NUMBER OF BLOCKS          *)
 ORCNUMBER    = 6;   (* NUMBER OF ORCS IN CAVERN       *)
 ORCSAFE      = 30;  (* MOVES BETWEEN ORC APPEARANCES  *)

 WORD          = PACKED ARRAY [ 1 .. WORDSIZE ] OF CHAR;
 LINE          = PACKED ARRAY [ 1 .. LINEWIDTH ] OF CHAR;
 DESCRIPTION = RECORD CASE NDLS : DESCNUM OF
                  1  : ( C1  : LINE );
                  2  : ( C2  : ARRAY [ 1 ..  2 ] OF LINE );
                  3  : ( C3  : ARRAY [ 1 ..  3 ] OF LINE );
                  4  : ( C4  : ARRAY [ 1 ..  4 ] OF LINE );
                  5  : ( C5  : ARRAY [ 1 ..  5 ] OF LINE );
                  6  : ( C6  : ARRAY [ 1 ..  6 ] OF LINE );
                  7  : ( C7  : ARRAY [ 1 ..  7 ] OF LINE );
                  8  : ( C8  : ARRAY [ 1 ..  8 ] OF LINE );
                  9  : ( C9  : ARRAY [ 1 ..  9 ] OF LINE );
                  10 : ( C10 : ARRAY [ 1 .. 10 ] OF LINE );
               END;
 DEFNTYPE      = ( DEFN,       NODE );
 LOCALE        = ( INSIDE,     OUTSIDE );
 QUESTTYPE     = ( NOQUEST,    INFOQUEST,  DEADQUEST,
                   LASTCHANCE,             QUITQUEST );
 WORDTYPE      = ( DIRECT,     ACT,        KNOWN,      LOCATE,
                   UNKNOWN );
 DIRECTION     = ( ENTER,      EXIT,       ALTER,      CROSS,
                   DOWN,       EAST,       JUMP,       MAGIC,
                   NORTH,      NORTHEAST,  NORTHWEST,  SOUTH,
                   SOUTHEAST,  SOUTHWEST,  UP,         WEST );
ACTION         = ( BRIEF,      BUILD,      DESCRIBE,   DRINK,
                   DROP,       EAT,        EMPTY,      FEED,
                   FILL,       HELP,       INFO,       INVEN,
                   KILL,       LEFT,       LOCK,       LOOK,
                   NO,         OFF,        ON,         QUIT,
                   RAISE,      RESIGN,     RIGHT,      ROW,     RUB,
                   SAVE,       SCORE,      SWIM,       TAKE,
                   THROW,      UNLOCK,     VERBOSE,    WAVE,
                   YES );
 NOUN          = ( NILL,       ALL,        AXE,        BOAT,
                   BOTTLE,     BRIDGE,     CAGE,       FOOD,
                   HAMMER,     KEYS,       KNIFE,      LADDER,
                   LAMP,       MATCH,      NAIL,       OIL,
                   PLANT,      ROD,        ROPE,       SHARD,
                   WATER,      WOOD,       CHAIN,      CHEST,
                   COIN,       CRYSTAL,    DIAMOND,    EGG,
                   EMERALD,    FUR,        GOLD,       IVORY,
                   NECKLACE,   PEARL,      PILLOW,     PLATINUM,
                   PYRAMID,    RING,       RUBY,       RUG,
                   SILVER,     SPICE,      TEAK,       TRIDENT,
                   VASE,       BEAR,       BIRD,       CLAM,
                   DRAGON,     ORC,     PIRATE,    SNAKE,
                   TROLL,      WOLF );
 NOUNSET       = SET OF AXE .. WOLF;
 LOCPTR        = ↑ LOCATION;
 DESCPTR       = ↑ DESCRIPTION;
 NAMEPTR       = ↑ NAMEDEFN;
 LOCATION      = PACKED RECORD
                   NAME    : NAMEPTR;
                   TELL    : DESCPTR;
                   PRESENT : NOUNSET;
                   PASSAGE : PACKED ARRAY [ ENTER .. WEST ] OF
                               PACKED RECORD
                                 GATE    : BOOLEAN;
```

```
                        TARGET : LOCPTR;
                      END;
               WET      : ( DRY, OILWET, WATERWET );
               CLASS    : ( LAND, BEACH, LAKE );
               POOL     : ( NOPOOL, OILPOOL, WATERPOOL );
               SIDE     : LOCALE;
               BLOCK    : BOOLEAN;
               WARN     : O .. NOBLOCKS;
               MAGCH    : CHAR;
               VISIT    : BOOLEAN;
             END;
      NAMEDEFN     = RECORD
               SLNK : NAMEPTR;
               CASE NDF : DEFNTYPE OF
                 DEFN : ( SDFN : WORD;
                          CASE MEANING : WORDTYPE OF
                            DIRECT  : ( DIRVAL   : DIRECTION );
                            ACT     : ( ACTVAL   : ACTION );
                            KNOWN   : ( NOUNVAL  : NOUN );
                            LOCATE  : ( LOCVAL   : LOCPTR );
                            UNKNOWN : ( UNVAL    : WORD ));
                 NODE : ( LLNK, RLNK : NAMEPTR );
             END;
      NOUNDESCR    = PACKED RECORD
               NAME      : NAMEPTR;
               TELL      : DESCPTR;
               TOLD      : BOOLEAN;
               SPECIAL   : BOOLEAN;
               CASE WHAT : NOUN OF
                 BOTTLE : ( BOTTLECONTENTS : ( EMPTYBOTTLE,
                                               OILINBOTTLE,
                                               WATERINBOTTLE ));
                 CAGE   : ( CAGECONTENTS   : ( EMPTYCAGE,
                                               BIRDINCAGE ));
                 LAMP   : ( BURNING        :   BOOLEAN;
                            TIMELEFT       :   INTEGER );
                 MATCH  : ( NOFMATCHES     :   INTEGER );
                 NAIL   : ( NOFNAILS       :   INTEGER );
                 PLANT  : ( HEIGHT         : ( LITTLE,
                                               STAGE1,
                                               STAGE2,
                                               FULL,
                                               OVERGROWN ));
                 WOOD   : ( PILESIZE       :   INTEGER );
                 CHEST  : ( CHESTSTATE     : ( EMPTYCHEST,
                                               TREASURECHEST,
                                               LOCKEDCHEST );
                            CHESTCONTENTS  :   NOUNSET );
                 BEAR   : ( BEARSTATE      : ( ANGRY,
                                               HAPPY ));
                 BIRD   : ( BIRDSTATE      : ( FREEBIRD,
                                               CAGEDBIRD ));
                 CLAM   : ( CLAMOPN        : ( NEVER,
                                               HASBEEN ));
                 WOLF   : ( LIFE           : ( ALIVE,
                                               DEAD ));
               END;
  *$N VARIABLE DEFINITIONS                                       *)

      NOUNS      : ARRAY [ ALL .. WOLF ] OF NOUNDESCR;
      STRING     : WORD;        (* WORD RETURNED BY READTOKEN      *)
      RVALUE      : INTEGER;     (* VALUE RETURNED BY READTOKEN     *)
      COLUMN     : INTEGER;     (* CURRENT DATA LINE COLUMN        *)
      DEFINITION : BOOLEAN;     (* INITIALIZATION OR EXECUTION FLAG *)
      LINEBUFFER : LINE;          (* ONE TEXT LINE FROM DATA OR TTY   *)
      ROOT       : NAMEPTR;       (* ROOT OF DATA STRUCTURE           *)
      VOID       : NAMEPTR;       (* UNKNOWN WORD DESCRIPTOR POINTER  *)
      WALL       : NAMEPTR;       (* POINTER TO 'ALL' DESCRIPTOR      *)
      WORDPTR    : NAMEPTR;       (* CURRENT WORD DEFINITION POINTER  *)
      COMMANDS   : INTEGER;       (* COUNT OF COMMANDS GIVEN          *)
      MOVES      : INTEGER;       (* COUNT OF MOVES TAKEN             *)
      DARKMOVES  : INTEGER;       (* COUNT OF MOVES IN THE DARK       *)
      DONE       : BOOLEAN;       (* FLAG WHEN ADVENTURER IS DONE     *)
      WHERE      : LOCPTR;        (* CURRENT LOCATION OF ADVENTURER   *)
      STARTLOC   : LOCPTR;        (* STARTING LOCATION OF ADVENTURE   *)
      WAS        : LOCPTR;        (* WHERE ADVENTURER LAST WAS        *)
      TREASLOC   : LOCPTR;        (* WHERE TREASURE MUST BE DEPOSITED *)
      MAXTREAS   : NOUNSET;       (* SET OF TREASURE PLACED IN CAVERN *)
      TREAS      : NOUNSET;       (* SET OF ALL POSSIBLE TREASURES    *)
      DISCOVT    : NOUNSET;       (* SET OF TREASURES DISCOVERED      *)
      QUESTION   : QUESTTYPE;     (* CURRENT QUESTION                 *)
      CARRY      : NOUNSET;       (* OBJECTS CARRIED BY ADVENTURER    *)
      BRIEFLY    : BOOLEAN;       (* BRIEF DESCRIPTIONS FLAG          *)
      NUMDIED    : INTEGER;       (* COUNT OF TIMES ADVENTURER DIED   *)
      ROWFLAG    : BOOLEAN;       (* WORD 'ROW' ENCOUNTERED           *)
      BURNTIME   : INTEGER;       (* TIME LAMP BURNS ON FUEL          *)
      NUMORC   : INTEGER;       (* NUMBER OF ORCS GENERATED      *)
      SAFORC   : INTEGER;       (* TIME BETWEEN ORC GENERATIONS   *)
      MAXVISIT   : INTEGER;       (* NUMBER OF POSSIBLE POSITIONS     *)
      ACTVISIT   : INTEGER;       (* ACTUAL POSITIONS VISITED         *)
      MOFDAY     : LINE;          (* MESSAGE OF THE DAY               *)
      STRANGLE   : INTEGER;       (* COUNT OF DRAGON/SNAKE CYCLES     *)
      BOATLOC    : LOCPTR;        (* HOME POSITION OF BOAT            *)
      BOATPOS    : LOCPTR;        (* CURRENT LOCATION OF BOAT         *)
      RIGHTECH   : BOOLEAN;       (* RIGHT TECHNIQUE FOR DRAGON       *)
      CHESTLOC   : LOCPTR;        (* LOCATION OF PIRATE'S CHEST       *)
      PSTATE     : ( PWAIT,
                     PACTIVE,
                     PDONE );     (* PIRATE STATES                    *)
*$N I/O PROCEDURES                                                 *)

*   PROCEDURE SNAP ( X : WORD );
 FORTRAN;      *)

ROCEDURE READLINE;
   VAR I, J : INTEGER;
       STOP : BOOLEAN;
 BEGIN (* READLINE *)
   I := 0;
   IF NOT DEFINITION THEN BEGIN
     IF EOF OR EOS THEN GETSEG(INPUT);
     IF EOLN THEN READLN;
     WHILE (NOT EOLN) AND (I < LINEWIDTH) DO BEGIN
       I := I + 1;
       READ(LINEBUFFER[I]); END; END
   ELSE BEGIN
     WHILE (NOT EOLN(ADVENT4)) AND (I < LINEWIDTH) DO BEGIN
       I := I + 1;
       READ(ADVENT4,LINEBUFFER[I]); END;
     READLN(ADVENT4); END;
   J := LINEWIDTH;
   REPEAT
     LINEBUFFER[J] := CHR(0);
     J := J - 1;
   UNTIL J <= I;
   STOP := FALSE;
   REPEAT
     IF LINEBUFFER[I] = ' ' THEN BEGIN
```

```
       LINEBUFFER[I] := CHR(0);
       I := I - 1;
       STOP := I < 1; END
     ELSE
       STOP := TRUE;
   UNTIL STOP;
   COLUMN := 1;
 END; (* READLINE *)

ROCEDURE READTOKEN;
   LABEL 1; (* UNKNOWN WORD RESCAN *)
   VAR DIGITS : BOOLEAN;
       CH     : CHAR;
       I, J   : INTEGER;
       LETTER,
       DIGIT,
       NULL   : SET OF COL .. '9';
 BEGIN (* READTOKEN *)
   LETTER := [ 'A' .. 'Z' ];
   DIGIT  := [ '0' .. '9' ];
   NULL   := [ CHR(0) ];
```

```
:  RVALUE := 0;
   DIGITS := TRUE;
   WHILE NOT (LINEBUFFER[COLUMN] IN (LETTER + DIGIT + NULL)) DO
     COLUMN := COLUMN + 1;
   I := 0;
   WHILE LINEBUFFER[COLUMN] IN (LETTER + DIGIT) DO BEGIN
     CH := LINEBUFFER[COLUMN];
     COLUMN := COLUMN + 1;
     DIGITS := DIGITS AND (CH IN DIGIT);
     IF I < WORDSIZE-1 THEN BEGIN
       I := I + 1;
       STRING[I] := CH; END; END;
   FOR J := I+1 TO WORDSIZE DO
     STRING[J] := CHR(0);
   IF STRING[1] <> CHR(0) THEN
     IF DIGITS THEN BEGIN
       IF NOT DEFINITION THEN
         GOTO 1;
       I := 1;
       WHILE STRING[I] <> CHR(0) DO BEGIN
         IF STRING[I] IN DIGIT THEN
           RVALUE := 10 * RVALUE + ORD(STRING[I]) - ORD('0');
         I := I + 1; END;
       STRING[1] := CHR(0); END
     ELSE BEGIN
       IF NOT DEFINITION THEN BEGIN
         WORDPTR := ROOT;
         WHILE WORDPTR↑.NDF <> DEFN DO
           IF WORDPTR↑.SLNK↑.SDFN < STRING THEN
             WORDPTR := WORDPTR↑.RLNK
           ELSE
             WORDPTR := WORDPTR↑.LLNK;
         IF WORDPTR↑.SDFN <> STRING THEN
           GOTO 1; END; END
   ELSE
     IF DEFINITION THEN
       WORDPTR := NIL
     ELSE
       WORDPTR := VOID;
 END; (* READTOKEN *)
*$N INITIALIZATION PROCEDURES                                   *)

UNCTION RANDOM(P: REAL): REAL; EXTERN;
```

```
ROCEDURE INITIALIZE;
   VAR DI : DIRECTION;
       AI : ACTION;
       NI : NOUN;
       I  : INTEGER;
       X  : REAL;

 PROCEDURE GETFIL; FORTRAN;
```

```
FUNCTION NEWDP ( N : INTEGER ) : DESCPTR;
    VAR DP : DESCPTR;
  BEGIN (* NEWDP *)
    CASE N OF
      1  : NEW(DP,1 );
      2  : NEW(DP,2 );
      3  : NEW(DP,3 );
      4  : NEW(DP,4 );
      5  : NEW(DP,5 );
      6  : NEW(DP,6 );
      7  : NEW(DP,7 );
      8  : NEW(DP,8 );
      9  : NEW(DP,9 );
      10 : NEW(DP,10);
    END;
    NEWDP := DP;
  END; (* NEWDP *)

FUNCTION INSERT : NAMEPTR;
    VAR P1, P2 : NAMEPTR;
        F      : BOOLEAN;
  BEGIN (* INSERT *)
    NEW(P1,DEFN);
    WITH P1↑ DO BEGIN
      SLNK := NIL;
      NDF  := DEFN;
      SDFN := STRING;
      MEANING := UNKNOWN; END;
    INSERT := P1;
    IF ROOT = NIL THEN
      ROOT := P1
    ELSE IF STRING < ROOT↑.SDFN THEN BEGIN
      P1↑.SLNK := ROOT;
      ROOT := P1; END
    ELSE BEGIN
      P2 := ROOT;
      F := P2↑.SLNK <> NIL;
      WHILE F DO BEGIN
        IF STRING > P2↑.SLNK↑.SDFN THEN
          P2 := P2↑.SLNK
        ELSE
          F := FALSE;
        F := F AND ( P2↑.SLNK <> NIL); END;
      P1↑.SLNK := P2↑.SLNK;
      P2↑.SLNK := P1; END;
  END; (* INSERT *)

FUNCTION FINDNAME ( T : WORDTYPE ) : NAMEPTR;
    VAR NP : NAMEPTR;
  BEGIN (* FINDNAME *)
    NP := ROOT;
    WHILE NP↑.SDFN <> STRING DO
      NP := NP↑.SLNK;
    IF T <> UNKNOWN THEN
      IF NP↑.MEANING <> T THEN
        HALT;
    FINDNAME := NP;
  END; (* FINDNAME *)

PROCEDURE NEXTTOKEN;
  BEGIN (* NEXTTOKEN *)
    REPEAT
      READTOKEN;
      IF STRING[1] = CHR(0) THEN
```

```
        READLINE;
      UNTIL STRING[1] <> CHR(0);
    END; (* NEXTTOKEN *)

  PROCEDURE ENTERDIRECTION ( D : DIRECTION );
      VAR P1 : NAMEPTR;
    BEGIN (* ENTERDIRECTION *)
      NEXTTOKEN;
      P1 := INSERT;
      P1↑.MEANING := DIRECT;
      P1↑.DIRVAL  := D;
    END; (* ENTERDIRECTION *)

  PROCEDURE ENTERACTION ( A : ACTION );
      VAR P1 : NAMEPTR;
    BEGIN (* ENTERACTION *)
      NEXTTOKEN;
      P1 := INSERT;
      P1↑.MEANING := ACT;
      P1↑.ACTVAL  := A;
    END; (* ENTERACTION *)

  PROCEDURE ENTERNOUN ( N : NOUN );
      VAR P1 : NAMEPTR;

    PROCEDURE READINT ( VAR INT : INTEGER );
        BEGIN (* READINT *)
          READTOKEN;
          IF (STRING[1] <> CHR(0)) OR (RVALUE = 0) THEN
            HALT;
          INT := RVALUE;
        END; (* READINT *)

    BEGIN (* ENTERNOUN *)
      NEXTTOKEN;
      P1 := INSERT;
      P1↑.MEANING := KNOWN;
      P1↑.NOUNVAL := N;
      WITH NOUNS[N] DO BEGIN
        NAME     := P1;
        TELL     := NIL;
        TOLD     := FALSE;
        WHAT     := N;
        SPECIAL := N IN [ BOTTLE, CAGE, LAMP, PLANT,
                          CHEST, BEAR, BIRD, CLAM, WOLF ];
        IF N IN [ BOTTLE, CAGE, LAMP, MATCH, NAIL, PLANT,
                  WOOD, CHEST, BEAR, BIRD, CLAM, WOLF ] THEN
          CASE N OF
            BOTTLE : BOTTLECONTENTS := WATERINBOTTLE;
            CAGE   : CAGECONTENTS := EMPTYCAGE;
            LAMP   : BEGIN
                       BURNING := FALSE;
                       READINT(BURNTIME);
                       TIMELEFT := BURNTIME; END;
            MATCH  : READINT(NOFMATCHES);
            NAIL   : READINT(NOFNAILS);
            PLANT  : HEIGHT := LITTLE;
            WOOD   : READINT(PILESIZE);
            CHEST  : BEGIN
                       CHESTSTATE := EMPTYCHEST;
                       CHESTCONTENTS := []; END;
            BEAR   : BEARSTATE := ANGRY;
            BIRD   : BIRDSTATE := FREEBIRD;
            CLAM   : CLAMOPN := NEVER;
          WOLF    : LIFE := ALIVE;
          END; END;
  END; (* ENTERNOUN *)

PROCEDURE LOADNOUNDESCRIPTIONS;
    VAR TEMP : DESCRIPTION;
        DP   : DESCPTR;
        NP   : NAMEPTR;
        I, J : INTEGER;
  BEGIN (* LOADNOUNDESCRIPTIONS *)
    WHILE ADVENT4↑ = '+' DO BEGIN
      READLINE;
      COLUMN := 2;
      READTOKEN;
      IF STRING[1] = CHR(0) THEN
        HALT;
      NP := FINDNAME(KNOWN);
      I := 0;
      WHILE (ADVENT4↑ <> '+') AND (ADVENT4↑ <> '-') DO BEGIN
        READLINE;
        I := I + 1;
        TEMP.C10[I] := LINEBUFFER; END;
      DP := NEWDP(I);
      DP↑.NDLS := I;
      FOR J := 1 TO I DO
        DP↑.C10[J] := TEMP.C10[J];
      NOUNS[NP↑.NOUNVAL].TELL := DP; END;
  END; (* LOADNOUNDESCRIPTIONS *)

PROCEDURE LOADSYNONYMS;
    VAR NP, SP, TP : NAMEPTR;
  BEGIN (* LOADSYNONYMS *)
    WHILE ADVENT4↑ = '-' DO BEGIN
      READLINE;
      COLUMN := 2;
      READTOKEN;
      NP := FINDNAME(UNKNOWN);
      READTOKEN;
      REPEAT
        SP := INSERT;
        TP := SP↑.SLNK;
        SP↑ := NP↑;
        SP↑.SLNK := TP;
        SP↑.SDFN := STRING;
        READTOKEN;
      UNTIL STRING[1] = CHR(0); END;
  END; (* LOADSYNONYMS *)

PROCEDURE LOADROOMDESCRIPTIONS;
    VAR TEMP : DESCRIPTION;
        DP   : DESCPTR;
        NP   : NAMEPTR;
        RP   : LOCPTR;
        DI   : DIRECTION;
        I, J : INTEGER;
  BEGIN (* LOADROOMDESCRIPTIONS *)
    WHILE ADVENT4↑ = '+' DO BEGIN
      I := 0;
      READLINE;
      NEW(RP);
      WITH RP↑ DO BEGIN
        NAME := NIL;
        TELL := NIL;
        PRESENT := [];
        FOR DI := ENTER TO WEST DO BEGIN
          PASSAGE[DI].GATE := FALSE;
          PASSAGE[DI].TARGET := NIL; END;
        WET    := DRY;
        CLASS  := LAND;
        POOL   := NOPOOL;
        SIDE   := INSIDE;
        BLOCK  := FALSE;
        WARN   := 0;
        VISIT  := FALSE;
        MAXVISIT := MAXVISIT + 1;
        IF LINEBUFFER[2] = 'B' THEN
          CLASS := BEACH;
        IF LINEBUFFER[2] = 'W' THEN
          CLASS := LAKE;
        IF LINEBUFFER[3] = 'O' THEN
          POOL := OILPOOL;
        IF LINEBUFFER[3] = 'W' THEN
          POOL := WATERPOOL;
        IF LINEBUFFER[4] = 'O' THEN
          SIDE := OUTSIDE;
        COLUMN := 5;
        READTOKEN;
        NP := INSERT;
        NAME := NP;
        WITH NP↑ DO BEGIN
          MEANING := LOCATE;
          LOCVAL := RP; END;
        I := 0;
        WHILE (ADVENT4↑ <> '-') AND (ADVENT4↑ <> '+') DO BEGIN
          READLINE;
          I := I + 1;
          TEMP.C10[I] := LINEBUFFER; END;
        DP := NEWDP(I);
        DP↑.NDLS := I;
        FOR J := 1 TO I DO
          DP↑.C10[J] := TEMP.C10[J];
        RP↑.TELL := DP; END; END;
  END; (* LOADROOMDESCRIPTIONS *)

PROCEDURE LOADINTERCONNECTIONS;
    VAR FP, TP, DP : NAMEPTR;
  BEGIN (* LOADINTERCONNECTIONS *)
    WHILE ADVENT4↑ = '-' DO BEGIN
      READLINE;
      COLUMN := 2;
      READTOKEN;
      FP := FINDNAME(LOCATE);
      READTOKEN;
      TP := FINDNAME(LOCATE);
      STRING[1] := CHR(0);
      REPEAT
        IF STRING[1] = CHR(0) THEN
          READTOKEN;
        IF STRING[1] <> CHR(0) THEN WITH FP↑.LOCVAL↑ DO BEGIN
          DP := FINDNAME(DIRECT);
          PASSAGE[DP↑.DIRVAL].TARGET := TP↑.LOCVAL;
          IF DP↑.DIRVAL = MAGIC THEN
            MAGCH := STRING[1];
          READTOKEN;
          IF (STRING[1] = CHR(0)) AND (RVALUE <> 0) THEN BEGIN
            PASSAGE[DP↑.DIRVAL].GATE := TRUE;
            BLOCK := RVALUE IN [ 1 .. 3, 5 .. 9, 11 ];
            WARN  := RVALUE; END; END;
      UNTIL (STRING[1] = CHR(0)) AND (RVALUE = 0); END;
  END; (* LOADINTERCONNECTIONS *)

PROCEDURE LOADNOUNLOCATIONS;
    VAR NP, LP : NAMEPTR;
  BEGIN (* LOADNOUNLOCATIONS *)
    WHILE ADVENT4↑ = '+' DO BEGIN
      READLINE;
      COLUMN := 2;
      READTOKEN;
      LP := FINDNAME(LOCATE);
      READTOKEN;
      REPEAT
        NP := FINDNAME(KNOWN);
        WITH NP↑ DO BEGIN
          IF NOUNVAL IN TREAS THEN
            MAXTREAS := MAXTREAS + [NOUNVAL];
          IF NOUNVAL = CLAM THEN
            MAXTREAS := MAXTREAS + [PEARL]; END;
        WITH LP↑.LOCVAL↑ DO
          PRESENT := PRESENT + [NP↑.NOUNVAL];
        READTOKEN;
      UNTIL STRING[1] = CHR(0); END;
  END; (* LOADNOUNLOCATIONS *)

PROCEDURE LOADSPECIALLOCATIONS;

  FUNCTION NEXT : LOCPTR;
      VAR NP : NAMEPTR;
    BEGIN (* NEXT *)
      READLINE;
      COLUMN := 2;
      READTOKEN;
      NP := FINDNAME(LOCATE);
      NEXT := NP↑.LOCVAL;
    END; (* NEXT *)

  BEGIN (* LOADSPECIALLOCATIONS *)
    STARTLOC := NEXT;
    TREASLOC := NEXT;
    BOATLOC  := NEXT;
    CHESTLOC := NEXT;
  END; (* LOADSPECIALLOCATIONS *)

PROCEDURE REMOVEWORDS;
    VAR P1, P2 : NAMEPTR;
  BEGIN (* REMOVEWORDS *)
    P1 := ROOT;
    P2 := NIL;
    REPEAT
      IF (P1↑.MEANING = LOCATE) OR
         ((P1↑.MEANING = DIRECT) AND (P1↑.DIRVAL = ALTER)) THEN
        IF P2 = NIL THEN BEGIN
          P1 := ROOT↑.SLNK;
          ROOT := P1; END
        ELSE BEGIN
          P1 := P1↑.SLNK;
          P2↑.SLNK := P1; END
      ELSE BEGIN
        P2 := P1;
        P1 := P1↑.SLNK; END;
    UNTIL P1 = NIL;
  END; (* REMOVEWORDS *)
```

```
PROCEDURE BUILDTREE;
    VAR P1, P2, P3 : NAMEPTR;

    FUNCTION TIESLNK ( VAR P1 : NAMEPTR ) : NAMEPTR;
      BEGIN (* TIESLNK *)
        IF P1↑.NDF = DEFN THEN
          TIESLNK := P1
        ELSE BEGIN
          P1↑.SLNK := TIESLNK(P1↑.LLNK);
          TIESLNK  := TIESLNK(P1↑.RLNK); END;
      END; (* TIESLNK *)

  BEGIN (* BUILDTREE *)
    WHILE ROOT↑.SLNK <> NIL DO BEGIN
      P1 := ROOT;
      NEW(P2,NODE);
      WITH P2↑ DO BEGIN
        SLNK := NIL;
        NDF  := NODE;
        LLNK := P1;
        RLNK := P1↑.SLNK; END;
      P1 := P1↑.SLNK↑.SLNK;
      ROOT := P2;
      WHILE P1 <> NIL DO
        IF P1↑.SLNK = NIL THEN BEGIN
          P2↑.SLNK := P1;
          P1 := NIL; END
        ELSE BEGIN
          NEW(P3,NODE);
          WITH P3↑ DO BEGIN
            SLNK := NIL;
            NDF  := NODE;
            LLNK := P1;
            RLNK := P1↑.SLNK; END;
          P1 := P1↑.SLNK↑.SLNK;
          P2↑.SLNK := P3;
          P2 := P3; END; END;
    P1 := TIESLNK(ROOT);
  END; (* BUILDTREE *)

BEGIN (* INITIALIZE *)
GETFIL;
  LINELIMIT(OUTPUT,-1);
  DEFINITION := TRUE;
  ROOT       := NIL;
  STRING[1]  := CHR(0);
  TREAS     := [ CHAIN .. VASE ];
  MAXTREAS := [];
  DISCOVT  := [];
  MAXVISIT := 0;
  ACTVISIT := 0;
  RESET(ADVENT4);
  READLINE;
  MOFDAY := LINEBUFFER;
  READLINE;
  FOR DI := ENTER TO WEST DO
    ENTERDIRECTION(DI);
  FOR AI := BRIEF TO YES DO
    ENTERACTION(AI);
  FOR NI := ALL TO WOLF DO
    ENTERNOUN(NI);
  LOADNOUNDESCRIPTIONS;
  LOADSYNONYMS;
  LOADROOMDESCRIPTIONS;
```

```
    LOADINTERCONNECTIONS;
    LOADNOUNLOCATIONS;
    LOADSPECIALLOCATIONS;
    BOATPOS           := BOATLOC;
    BOATLOC↑.PRESENT := BOATLOC↑.PRESENT + [BOAT];
    REMOVEWORDS;
    BUILDTREE;
    WHERE := STARTLOC;
    WAS   := NIL;
    NEW(VOID,DEFN,UNKNOWN);
    WITH VOID↑ DO BEGIN
      SLNK    := NIL;
      NDF     := DEFN;
      MEANING := UNKNOWN; END;
    NEW(WALL,DEFN,KNOWN);
    WITH WALL↑ DO BEGIN
      SLNK    := NIL;
      NDF     := DEFN;
      MEANING := KNOWN;
      NOUNVAL := ALL; END;
    STRING[1]  := CHR(0);
    RVALUE      := 0;
    COLUMN     := 1;
    DEFINITION := FALSE;
    WORDPTR    := NIL;
    COMMANDS   := 0;
    MOVES      := 0;
    DARKMOVES  := 0;
    DONE       := FALSE;
    BRIEFLY    := FALSE;
    QUESTION   := INFOQUEST;
    CARRY      := [];
    NUMDIED    := 0;
    ROWFLAG    := FALSE;
    NUMORC   := ORCNUMBER;
    SAFORC   := ORCSAFE;
    STRANGLE   := 0;
    RIGHTECH   := FALSE;
    PSTATE     := PWAIT;
*     SNAP('ADVORG               ');          *)
    IF MOFDAY[1] <> CHR(0) THEN BEGIN
      I := 1;
      WHILE MOFDAY[I] <> CHR(0) DO BEGIN
        WRITE(MOFDAY[I]);
        I := I + 1; END;
        WRITELN;
        WRITELN; END;
    WRITELN('WELCOME TO ADVENTURE!  WOULD YOU LIKE INSTRUCTIONS?');
    FOR I := 1 TO (CLOCK MOD 100) DO
      X := RANDOM(0);
 END; (* INITIALIZE *)
*$N UTILITY PROCEDURES                                          *)

ROCEDURE PRINT ( VAR L : LINE );
   VAR I : INTEGER;
 BEGIN (* PRINT *)
   I := 1;
   WHILE L[I] <> CHR(0) DO BEGIN
     WRITE(L[I]);
     I := I + 1; END;
   WRITELN;
 END; (* PRINT *)

UNCTION PWORD : CHAR;
```

```
    VAR I : INTEGER;
        C : CHAR;
        F : BOOLEAN;
  BEGIN (* PWORD *)
    I := 1;
    REPEAT
      C := STRING[I];
      I := I + 1;
      F := I < 11;
      IF F THEN
        F := STRING[I] <> CHR(0);
      IF F THEN
        WRITE(C);
    UNTIL NOT F;
    PWORD := C;
  END; (* PWORD *)

ROCEDURE TELLNOUN ( N : NOUN; BREVITY : BOOLEAN );
    VAR I     : INTEGER;
        BRIEF : BOOLEAN;
        CN    : NOUN;
  BEGIN (* TELLNOUN *)
    BRIEF := BREVITY;
    WITH NOUNS[N], TELL↑ DO BEGIN
      IF NOT TOLD THEN BEGIN
        IF N IN TREAS THEN
          DISCOVT := DISCOVT + [N];
        TOLD := TRUE;
        BRIEF := FALSE; END;
      IF BRIEF THEN
        PRINT(C10[1])
      ELSE
        IF SPECIAL THEN
          CASE WHAT OF
            BOTTLE: PRINT(C10[ORD(BOTTLECONTENTS)+2]);
            CAGE  : PRINT(C10[ORD(CAGECONTENTS)+2]);
            LAMP  : PRINT(C10[ORD(BURNING)+2]);
            PLANT : PRINT(C10[ORD(HEIGHT)+2]);
            CHEST : BEGIN
                      PRINT(C10[ORD(CHESTSTATE)+2]);
                      PSTATE := PDONE;
                      IF CHESTSTATE = TREASURECHEST THEN
                        IF CHESTCONTENTS <> [] THEN BEGIN
                          WRITE  ('  I SEE THE FOLLOWING ');
                          WRITELN('THINGS INSIDE:');
                          FOR CN := CHAIN TO VASE DO
                            IF CN IN CHESTCONTENTS THEN
                              TELLNOUN(CN,TRUE); END; END;
            BEAR  : PRINT(C10[ORD(BEARSTATE)+2]);
            BIRD  : PRINT(C10[ORD(BIRDSTATE)+2]);
            CLAM  : PRINT(C10[ORD(CLAMOPN)+2]);
            WOLF  : PRINT(C10[ORD(LIFE)+2]);
          END
        ELSE
          FOR I := 2 TO NDLS DO
            PRINT(C10[I]); END;
   END; (* TELLNOUN *)
```

```
ROCEDURE OOPS;
 BEGIN (* OOPS *)
   ROWFLAG := FALSE;
   IF WORDPTR↑.MEANING = DIRECT THEN
     MOVES := MOVES + 1;
   WRITELN;
   IF NUMDIED = O THEN
     WRITELN('MY!  YOU SEEM TO HAVE GOTTEN YOURSELF KILLED!')
   ELSE
     WRITELN('MY!  YOU SEEM TO HAVE GOTTEN YOURSELF KILLED AGAIN!');
   CASE NUMDIED OF
     O : BEGIN
           WRITELN('BEING MAGICAL, I MAY BE ABLE TO HELP YOU.');
           WRITELN('DO YOU WANT ME TO TRY TO REVERSE THE EFFECTS');
           WRITELN('OF YOUR RECENT DEATH?'); END;
     1 : BEGIN
           WRITELN('YOU CERTAINLY KNOW HOW TO GET INTO TROUBLE!');
           WRITELN('THE SECOND TIME, REVERSING THE EFFECTS OF');
           WRITELN('DEATH IS HARDER.  SHALL I TRY?'); END;
     2 : BEGIN
           WRITELN('WELL!  THAT DOES IT!  I''M NOT GOING TO HELP');
           WRITELN('YOU IF YOU CAN''T STAY OUT OF TROUBLE.  DO');
           WRITELN('YOU WANT TO TRY YOUR OWN RESURRECTION?'); END;
     3 : BEGIN
           WRITELN('TAPS OLD BUDDY.  SO LONG!');
          DONE:=TRUE; END;
   END;
   NUMDIED := NUMDIED + 1;
   QUESTION := DEADQUEST;
   GOTO 50; (* ASK QUESTION IMMEDIATELY *)
 END; (* OOPS *)

UNCTION LIGHT : BOOLEAN;
```

```
 BEGIN (* LIGHT *)
   WITH WHERE↑, NOUNS[LAMP] DO
     LIGHT := (SIDE = OUTSIDE) OR
              ((LAMP IN (CARRY + PRESENT)) AND BURNING);
 END; (* LIGHT *)

ROCEDURE WARNING;
 BEGIN (* WARNING *)
   WITH WHERE↑ DO
     IF BLOCK THEN BEGIN
```

```
          IF WARN IN [ 1 .. 9 ] THEN
            CASE WARN OF
               1: WRITELN('THE DOOR IS LOCKED.');
               2: WRITELN('THE GRATE IS LOCKED.');
               3: WRITELN('THE IRON DOOR IS RUSTED SHUT.');
               4: TELLNOUN(PLANT,FALSE);
               5: WRITELN('THE FISSURE BLOCKS YOUR WAY.');
               6: WRITELN('A SHIMMERING WALL BLOCKS THE PASSAGE.');
               7: WRITELN('THE FAULT BLOCKS YOUR WAY.');
               8: WRITELN('THE WALL BLOCKS YOUR WAY.');
               9: WRITELN('THE CANYON BLOCKS YOUR WAY.');
            END; END
        ELSE
          IF WARN IN [ 1 .. 9, 12 ] THEN
            CASE WARN OF
               1: WRITELN('THE DOOR IS UNLOCKED.');
               2: WRITELN('THE GRATE IS UNLOCKED.');
               3: WRITELN('THE IRON DOOR IS OPEN.');
               4: TELLNOUN(PLANT,FALSE);
               5: WRITELN('A CRYSTAL BRIDGE SPANS THE FISSURE.');
               6: WRITELN('THE PASSAGE IS CLEAR.');
               7: WRITELN('THE ROPE PROVIDES A WAY PAST THE FAULT.');
               8: WRITELN('THE LADDER RESTS IN PLACE.');
               9: WRITELN('THE BRIDGE SPANS THE CANYON.');
              12: IF (COMMANDS MOD 3) <> 0 THEN
                    WRITELN('AN ARMED GUARD PATROLS THE PASSAGE.');
            END;
    END; (* WARNING *)

   ROCEDURE TELLLOCATION ( BREVITY : BOOLEAN );
      VAR I     : INTEGER;
          NP    : NOUN;
          TSET  : NOUNSET;
    BEGIN (* TELLLOCATION *)
      IF LIGHT THEN WITH WHERE↑, TELL↑ DO BEGIN
        IF BREVITY AND VISIT THEN
          PRINT(C10[1])
        ELSE
          FOR I := 2 TO NDLS DO
            PRINT(C10[I]);
        IF NOT VISIT THEN BEGIN
          ACTVISIT := ACTVISIT + 1;
          VISIT := TRUE; END;
        CASE WET OF
          DRY      : ;
          OILWET   : WRITELN('THE GROUND IS WET WITH OIL.');
          WATERWET : WRITELN('THE GROUND IS WET WITH WATER.');
        END;
        IF WARN <> 0 THEN
          WARNING;
        IF BOAT IN CARRY THEN
          WRITELN('YOU''RE IN A BOAT.');
        IF BEAR IN CARRY THEN
          WRITELN('A LARGE BEAR IS FOLLOWING YOU.');
        TSET := PRESENT - [OIL,PLANT,WATER,DRAGON,SNAKE];
        IF WARN IN [ 7, 8, 9 ] THEN
          CASE WARN OF
            7: TSET := TSET - [ROPE];
            8: TSET := TSET - [LADDER];
            9: TSET := TSET - [BRIDGE];
          END;
        IF TSET <> [] THEN BEGIN
          IF CARD(TSET) <= 1 THEN
            WRITELN('  I SEE AN OBJECT HERE.')
        ELSE
          WRITELN('  I SEE OBJECTS HERE.');
        FOR NP := AXE TO WOLF DO
          IF NP IN TSET THEN
            TELLNOUN(NP,TRUE); END;
      IF DRAGON IN PRESENT THEN
        WRITELN('THE DRAGON BLOCKS YOUR WAY!');
      IF WOLF   IN PRESENT THEN
        WRITELN('THE WOLF BLOCKS YOUR WAY!');
      IF BEAR IN PRESENT THEN
        WRITELN('THE BEAR BLOCKS YOUR WAY!');
      IF SNAKE IN PRESENT THEN
        WRITELN('THE SNAKE BLOCKS YOUR WAY!'); END
    ELSE BEGIN
      WRITELN('ITS TOO DARK TO SEE WHERE YOU''RE GOING.  IF YOU');
      WRITELN('PROCEED MUCH FURTHER YOU MAY FALL INTO A PIT.'); END;
  END; (* TELLLOCATON *)

ROCEDURE SCOREGAME;
   VAR S, T : INTEGER;
 BEGIN (* SCOREGAME *)
   S := TRUNC(  70
              + 100 * CARD(DISCOVT)/CARD(MAXTREAS)
              + 200 * CARD(MAXTREAS*TREASLOC↑.PRESENT)/CARD(MAXTREAS)
              + 130 * ACTVISIT/MAXVISIT
              -  17 * NUMDIED );
   WRITELN('YOU MOVED      ',   MOVES:4,' TIMES.');
   WRITELN('YOU GAVE ME    ',COMMANDS:4,' COMMANDS.');
   WRITELN('YOUR SCORE WAS ',       S:4,' OUT OF 500.');
   IF DONE THEN BEGIN
     WRITELN;
     IF S = 500 THEN
       T := 6
     ELSE BEGIN
       T := S DIV 85;
       S := 85 * (T + 1) - S;
       IF T = 5 THEN
         S := S - 10; END;
     CASE T OF
       0: WRITELN('YOU ARE OBVIOUSLY A RANK AMATEUR.');
       1: WRITELN('YOU''VE ACHIEVED THE RANK OF NOVICE ADVENTURER.');
       2: WRITELN('YOU''VE ACHIEVED THE RANK OF JUNIOR ADVENTURER.');
       3: WRITELN('YOU''RE A MASTER ADVENTURER, CLASS A.');
       4: WRITELN('YOU''RE A MASTER ADVENTURER, CLASS B.');
       5: WRITELN('YOU''RE A MASTER ADVENTURER, CLASS C.');
       6: WRITELN('YOU''RE A GRANDMASTER ADVENTURER!');
     END;
     IF T <> 6 THEN
       WRITELN('YOU NEED ',S:1,' MORE POINTS FOR THE NEXT RANK.');
     IF T = 5 THEN BEGIN
       WRITELN('YOU LOST POINTS FOR THE FOLLOWING REASON(S):');
       IF DISCOVT <> MAXTREAS THEN
         WRITELN('NOT FINDING ALL TREASURES.');
       IF DISCOVT*TREASLOC↑.PRESENT <> DISCOVT THEN
         WRITELN('NOT TAKING TREASURES TO THE PROPER PLACE.');
       IF ACTVISIT <> MAXVISIT THEN
         WRITELN('NOT COMPLETELY EXPLORING THE PARK.');
       IF NUMDIED <> 0 THEN
         WRITELN('DYING OR RESIGNING.'); END; END;
 END; (* SCOREGAME *)
*$N GAME PLAYING PROCEDURES                                            *)

ROCEDURE QUERYHUMAN;
   LABEL 99; (* DON''T UNDERSTAND LOOP *)
  VAR   FIRST : BOOLEAN;

PROCEDURE BLEWIT;
  BEGIN (* BLEWIT *)
    ROWFLAG := FALSE;
    CASE TRUNC(2.999*RANDOM(0)) OF
      0: WRITELN('I DON''T UNDERSTAND.');
      1: WRITELN('PLEASE REPHRASE THAT.');
      2: WRITELN('YOU''VE GOT TO BE KIDDING.');
    END;
    GOTO 99; (* DON''T UNDERSTAND LOOP *)
  END; (* BLEWIT *)

PROCEDURE DOMOVEMENT;
    VAR GO   : BOOLEAN;
        LAST : LOCPTR;
  BEGIN (* DOMOVEMENT *)
    WITH WHERE↑ DO
      IF ([DRAGON,SNAKE,BEAR,WOLF] * PRESENT) <> [] THEN BEGIN
        IF DRAGON IN PRESENT THEN
          WRITELN('THE DRAGON BLOCKS YOUR WAY!')
        ELSE IF (WOLF IN PRESENT) AND (ALIVE =  NOUNS[WOLF].LIFE) THEN
          WRITELN('THE WOLF BLOCKS YOUR WAY!')
        ELSE IF (BEAR IN PRESENT) AND
                (ANGRY =  NOUNS[BEAR].BEARSTATE) THEN
          WRITELN('THE BEAR BLOCKS YOUR WAY!')
        ELSE (* THE SNEAKY SNAKE *)
          WRITELN('THE SNAKE BLOCKS YOUR WAY!');
        GO := FALSE; END
      ELSE
        CASE CLASS OF
          LAND  : IF ROWFLAG THEN BEGIN
                    WRITE  ('YOU CAN''T ROW ON LAND, ');
                    WRITELN('YOU''RE NOT EVEN IN A BOAT.');
                    GO := FALSE; END
                  ELSE
                    GO := TRUE;
          BEACH : IF NOT (WORDPTR↑.DIRVAL IN [ENTER,EXIT]) THEN
                    IF ROWFLAG AND (NOT (BOAT IN CARRY)) THEN BEGIN
                      WRITELN('YOU CAN''T RW ON LAND.');
                      GO := FALSE; END
                    ELSE IF (NOT ROWFLAG) AND
                            (BOAT IN CARRY) THEN BEGIN
                      WRITELN('YOU MUST ROW YOUR BOAT.');
                      GO := FALSE; END
                    ELSE
                      GO := TRUE
                  ELSE
                    GO := TRUE;
          LAKE  : IF ROWFLAG OR
                     (WORDPTR↑.DIRVAL IN [ENTER,EXIT]) THEN
                    GO := TRUE
                  ELSE BEGIN
                    WRITE  ('YOU CAN''T WALK ON WATER!  ');
                    WRITELN('STAY IN YOUR BOAT.');
                    GO := FALSE; END;
        END;
    WHILE GO DO BEGIN
      LAST := WHERE;
      WITH WORDPTR↑, WHERE↑ DO
        IF MEANING <> DIRECT THEN
          BLEWIT
        ELSE IF (DIRVAL IN [ENTER,EXIT]) AND
```

```
            (BOAT IN CARRY+PRESENT) THEN
         IF DIRVAL = ENTER THEN
           IF BOAT IN CARRY THEN BEGIN
             GO := FALSE;
             WRITELN('YOU''RE ALREADY IN THE BOAT.'); END
           ELSE BEGIN
             CARRY := CARRY + [BOAT];
             PRESENT := PRESENT - [BOAT];
             WRITELN('YOU''VE LAUNCHED THE BOAT.'); END
         ELSE
           IF BOAT IN CARRY THEN
             IF CLASS = BEACH THEN BEGIN
               CARRY := CARRY - [BOAT];
               PRESENT := PRESENT + [BOAT];
               BOATPOS := WHERE;
               WRITELN('YOU''VE BEACHED THE BOAT.'); END
             ELSE BEGIN
               WRITELN('LEAVING THE BOAT HERE GETS YOU VERY WET');
               WRITELN('SINCE YOU CANNOT TREAD WATER VERY LONG.');
               OOPS; END
           ELSE BEGIN
             GO := FALSE;
             WRITELN('YOU''RE NOT IN THE BOAT.'); END
       ELSE IF (DIRVAL = JUMP) AND
               (WARN IN [ 5, 7 .. 9, 13, 17, 19 ]) THEN BEGIN
         WRITELN('YOU FALL TO YOUR DEATH.');
         OOPS; END
       ELSE WITH PASSAGE[DIRVAL] DO
         IF TARGET <> NIL THEN
           IF GATE THEN
             IF BLOCK THEN BEGIN
               WARNING;
               GO := FALSE; END
             ELSE BEGIN
               IF WARN IN [ 4, 12, 13, 15 .. 20 ] THEN
                 CASE WARN OF
                    4: IF GATE AND
                           (NOUNS[PLANT].HEIGHT <> FULL) THEN
                          WHERE := PASSAGE[ALTER].TARGET
                        ELSE
                          WHERE := TARGET;
                   12: IF (COMMANDS MOD 3) <> 0 THEN BEGIN
                          WRITE  ('THE ARMED GUARD SEES ');
                          WRITELN('YOU AND ATTACKS!');
                          OOPS; END
                        ELSE
                          WHERE := TARGET;
                   13: BEGIN
                          WRITELN('YOU HAVE FALLEN OFF THE CLIFF.');
                          OOPS; END;
                   15: BEGIN
                          WRITELN('THE CAVE CEILING FALLS ON YOU.');
                          OOPS; END;
                   16: IF CARRY <> [] THEN BEGIN
                          WRITE  ('YOU''RE CARRYING TOO');
                          WRITELN(' MUCH TO GO THRU.');
                          GO := FALSE; END
                        ELSE
                          WHERE := TARGET;
                   17: BEGIN
                          WRITE  ('YOU HAVE FALLEN DOWN ');
                          WRITELN('A BOTTOMLESS PIT.');
                          OOPS; END;
                   18: BEGIN
```

```
                     WRITE  ('YOU HAVE SUFFOCATED ');
                     WRITELN('IN BAD AIR.');
                     OOPS; END;
               19: IF (CARRY - [LAMP]) <> [] THEN BEGIN
                     WRITE  ('YOU''RE CARRYING SO MUCH ');
                     WRITELN('THAT YOU LOOSE YOUR');
                     WRITE  ('BALANCE ON THE LEDGE AND ');
                     WRITELN('FALL TO YOUR DEATH.');
                     OOPS; END
                   ELSE
                     WHERE := TARGET;
               20: IF RANDOM(0) < 0.5 THEN
                     WHERE := TARGET
                   ELSE
                     WHERE := PASSAGE[ALTER].TARGET;
            END
          ELSE
            WHERE := TARGET;
          IF (WHERE = WAS) AND GO THEN
            WAS := NIL; END
      ELSE BEGIN
        IF DIRVAL = MAGIC THEN
          IF STRING[1] <> MAGCH THEN
            BLEWIT;
        WHERE := TARGET;
        IF WHERE = WAS THEN
          WAS := NIL; END
    ELSE
      IF DIRVAL = MAGIC THEN
        BLEWIT
      ELSE BEGIN
        GO := FALSE;
        WRITELN('THERE IS NO WAY TO GO THAT DIRECTION.');
        IF WHERE <> WAS THEN
          WRITELN; END;
IF GO THEN
  MOVES := MOVES + 1;
CASE WHERE↑.CLASS OF
  LAND  : IF BOAT IN CARRY THEN BEGIN
            WRITELN('YOU''VE ROWED YOUR BOAT ONTO LAND.');
            WRITELN('THAT WAS SO HARD TO DO THAT');
            WRITELN('YOU GOT A HEART ATTACK.');
            WHERE := LAST;
            OOPS; END;
  BEACH : ;
  LAKE  : IF NOT (BOAT IN CARRY) THEN BEGIN
            WRITELN('YOU''VE WALKED OUT ON WATER');
            WRITELN('WHICH IS NOT KOSHER.');
            WHERE := LAST;
            OOPS; END;
END;
IF LIGHT THEN
  DARKMOVES := 0
ELSE BEGIN
  GO := FALSE;
  DARKMOVES := DARKMOVES + 1;
  IF DARKMOVES > 3 THEN
    IF RANDOM(0) < 0.10 THEN BEGIN
      WRITELN;
      WRITELN('YOU FELL INTO A PIT!');
      OOPS; END; END;
IF GO THEN BEGIN
  READTOKEN;
  GO := GO AND (WORDPTR↑.MEANING = DIRECT); END; END;
  END; (* DOMOVEMENT *)

PROCEDURE DOACTION;
    VAR AP : ACTION;
        WP : NAMEPTR;
        ST : WORD;

  PROCEDURE CHANGEBLOCK;
      VAR DI : DIRECTION;
    BEGIN (* CHANGEBLOCK *)
      WITH WHERE↑ DO BEGIN
        BLOCK := NOT BLOCK;
        DI := ENTER;
        WHILE NOT PASSAGE[DI].GATE DO
          DI := SUCC(DI);
        WITH PASSAGE[DI] DO
          IF TARGET↑.WARN = WARN THEN
            TARGET↑.BLOCK := NOT TARGET↑.BLOCK; END;
    END; (* CHANGEBLOCK *)

  PROCEDURE REMOVEBLOCK;
      VAR DI : DIRECTION;
    BEGIN (* REMOVEBLOCK *)
      WITH WHERE↑ DO BEGIN
        BLOCK := FALSE;
        DI := ENTER;
        WHILE NOT PASSAGE[DI].GATE DO
          DI := SUCC(DI);
        WITH PASSAGE[DI] DO
          IF TARGET↑.WARN = WARN THEN BEGIN
            TARGET↑.BLOCK := FALSE;
            IF WARN IN [ 7 .. 9 ] THEN
              CASE WARN OF
                7: TARGET↑.PRESENT := TARGET↑.PRESENT + [ROPE];
                8: TARGET↑.PRESENT := TARGET↑.PRESENT + [LADDER];
                9: TARGET↑.PRESENT := TARGET↑.PRESENT + [BRIDGE];
            END; END; END;
    END; (* REMOVEBLOCK *)

  PROCEDURE SETBLOCK;
      VAR DI : DIRECTION;
    BEGIN (* SETBLOCK *)
      WITH WHERE↑ DO BEGIN
        BLOCK := TRUE;
        DI := ENTER;
        WHILE NOT PASSAGE[DI].GATE DO
          DI := SUCC(DI);
        WITH PASSAGE[DI] DO
          IF TARGET↑.WARN = WARN THEN BEGIN
            TARGET↑.BLOCK := TRUE;
            IF WARN IN [ 7 .. 8 ] THEN
              CASE WARN OF
                7: TARGET↑.PRESENT := TARGET↑.PRESENT - [ROPE];
                8: TARGET↑.PRESENT := TARGET↑.PRESENT - [LADDER];
            END; END; END;
    END; (* SETBLOCK *)

  PROCEDURE ACTIONTAKE;
      VAR CP   : NOUN;
          GO   : BOOLEAN;
          TSET : NOUNSET;

    FUNCTION LOADOK ( N : NOUNSET ) : BOOLEAN;
        VAR WGHT : INTEGER;
        NV    : NOUN;
  BEGIN (* LOADOK *)
    WGHT := 0;
    FOR NV := AXE TO WOLF DO
      IF NV IN (CARRY + N) THEN
        IF NV IN [ AXE, BOTTLE, CAGE .. KEYS, LAMP .. NAIL,
                   PLANT .. ROPE, CHAIN, COIN .. VASE ] THEN
          WGHT := WGHT + 1
        ELSE IF NV = KNIFE THEN
          WGHT := WGHT + 2
        ELSE IF NV = LADDER THEN
          WGHT := WGHT + 3
        ELSE IF NV = CLAM THEN
          WGHT := WGHT + 8
        ELSE IF NV = WOOD THEN
          WGHT := WGHT + NOUNS[WOOD].PILESIZE
        ELSE IF NV = CHEST THEN WITH NOUNS[CHEST] DO
          IF (CHESTSTATE <> TREASURECHEST) OR
             (CHESTCONTENTS = []) THEN
            WGHT := WGHT + 1
          ELSE
            WGHT := WGHT + 10;
    LOADOK := WGHT <= 9;
  END; (* LOADOK *)

PROCEDURE DOTAKE;
  BEGIN (* DOTAKE *)
    WITH WHERE↑ DO
      IF CP IN [ OIL, WATER ] THEN BEGIN
        WRITE  ('YOU CAN''T TAKE ',PWORD,', IT WILL ');
        WRITELN('FLOW THROUGH YOUR HANDS.');
        GO := FALSE; END
      ELSE IF CP IN (PRESENT + TSET) THEN
        IF CP IN [ BOAT, PLANT, BRIDGE,
                   DRAGON .. WOLF ] THEN BEGIN
          WRITELN('YOU CAN''T TAKE A ',PWORD,'.');
          GO := FALSE; END
        ELSE IF CP = BEAR THEN
          IF NOUNS[BEAR].BEARSTATE = ANGRY THEN BEGIN
            WRITE  ('THE BEAR IS ANGRY AND DOES ');
            WRITELN('NOT LIKE YOUR APPROACH.');
            WRITELN('HE ATTACKS YOU AND MAULS YOU.');
            OOPS; END
          ELSE BEGIN
            PRESENT := PRESENT - [BEAR];
            CARRY := CARRY + [BEAR];
            WRITELN('THE BEAR WILL FOLLOW YOU.'); END
        ELSE IF CP = BIRD THEN
          IF ROD IN CARRY THEN BEGIN
            WRITE  ('THE BIRD WAS UNAFRAID WHEN ');
            WRITELN('YOU FIRST APPEARED,');
            WRITELN('BUT NOW IT WITHDRAWS AS YOU APPROACH.');
            GO := FALSE; END
          ELSE IF CAGE IN CARRY THEN BEGIN
            PRESENT := PRESENT - [BIRD];
            CARRY := CARRY + [BIRD];
            NOUNS[CAGE].CAGECONTENTS := BIRDINCAGE;
            NOUNS[BIRD].BIRDSTATE := CAGEDBIRD; END
          ELSE BEGIN
            WRITELN('YOU NEED A BIRD CAGE.');
            GO := FALSE; END
        ELSE
          IF LOADOK([CP]) THEN BEGIN
            IF CP IN PRESENT THEN
```

```
              PRESENT := PRESENT - [CP]
            ELSE WITH NOUNS[CHEST] DO
              CHESTCONTENTS := CHESTCONTENTS - [CP];
            CARRY := CARRY + [CP];
            IF CP IN TREAS THEN
              DISCOVT := DISCOVT + [CP];
            IF WARN IN [ 7, 8 ] THEN
              CASE WARN OF
                7: IF CP = ROPE THEN
                     SETBLOCK;
                8: IF CP = LADDER THEN
                     SETBLOCK;
              END;
            IF CP = CAGE THEN
              IF NOUNS[CAGE].CAGECONTENTS =
                 BIRDINCAGE THEN BEGIN
                PRESENT := PRESENT - [BIRD];
                CARRY   := CARRY   + [BIRD]; END; END
          ELSE BEGIN
            WRITE  ('YOU CAN''T CARRY ANY MORE ');
            WRITELN('WEIGHT.  YOU''LL HAVE');
            WRITELN('TO DROP SOMETHING FIRST.');
            GO := FALSE; END
        ELSE BEGIN
          IF LIGHT THEN
            WRITELN('I SEE NO ',PWORD,' HERE.')
          ELSE
            WRITELN('IT''S TOO DARK TO SEE ANYTHING.');
          GO := FALSE; END;
  END; (* DOTAKE *)

BEGIN (* ACTIONTAKE *)
  WITH WHERE↑ DO BEGIN
    TSET := PRESENT - [BOAT,BRIDGE,OIL,PLANT,
                       WATER,DRAGON..WOLF];
    IF CHEST IN PRESENT THEN WITH NOUNS[CHEST] DO
      IF CHESTSTATE = TREASURECHEST THEN
        TSET := TSET + CHESTCONTENTS;
    IF WORDPTR↑.NOUNVAL = ALL THEN
      IF NOT LIGHT THEN
        WRITELN('IT''S TOO DARK TO SEE ANYTHING.')
      ELSE IF TSET = [] THEN
        WRITELN('I SEE NOTHING TO TAKE.')
      ELSE IF NOT LOADOK(TSET) THEN
        WRITELN('YOU CAN''T TAKE EVERYTHING HERE.')
      ELSE BEGIN
        IF CARD(TSET) > 1 THEN
          WRITELN('YOU''VE TAKEN THE FOLLOWING THINGS:')
        ELSE
          WRITELN('YOU''VE TAKEN THE FOLLOWING THING:');
        FOR CP := AXE TO CLAM DO
          IF CP IN TSET THEN BEGIN
            GO := TRUE;
            DOTAKE;
            IF GO THEN
              TELLNOUN(CP,TRUE); END; END
    ELSE BEGIN
      GO := TRUE;
      REPEAT
        CP := WORDPTR↑.NOUNVAL;
        DOTAKE;
        READTOKEN;
      UNTIL (NOT GO) OR (WORDPTR↑.MEANING <> KNOWN);
      IF GO THEN
```

```
          WRITELN('OK.'); END; END;
  END; (* ACTIONTAKE *)

PROCEDURE ACTIONDROP;
    VAR CP   : NOUN;
        GO   : BOOLEAN;
        CLM  : (NOCM,ONECM,TWOCM);

  PROCEDURE DODROP;
    BEGIN (* DODROP *)
      WITH WHERE↑ DO
        IF CP IN [ OIL, WATER ] THEN BEGIN
          WRITE  ('DON''T BE SILLY, YOU POUR ');
          WRITELN(PWORD,', NOT DROP IT.');
          GO := FALSE; END
        ELSE IF CP = BIRD THEN
          IF BIRD IN CARRY THEN BEGIN
            NOUNS[CAGE].CAGECONTENTS := EMPTYCAGE;
            NOUNS[BIRD].BIRDSTATE := FREEBIRD;
            CARRY   := CARRY   - [BIRD];
            IF CLASS <> LAKE THEN
              PRESENT := PRESENT + [BIRD]; END
          ELSE BEGIN
            WRITELN('YOU''RE NOT CARRYING A BIRD.');
            GO := FALSE; END
        ELSE IF CP = CAGE THEN
          IF CAGE IN CARRY THEN
            IF BIRD IN CARRY THEN BEGIN
              CARRY   := CARRY   - [BIRD,CAGE];
              IF CLASS <> LAKE THEN
                PRESENT := PRESENT + [BIRD,CAGE]; END
            ELSE BEGIN
              CARRY   := CARRY   - [CAGE];
              IF CLASS <> LAKE THEN
                PRESENT := PRESENT + [CAGE]; END
          ELSE BEGIN
            WRITELN('YOU''RE NOT CARRYING A CAGE.');
            GO := FALSE; END
        ELSE IF CP IN CARRY THEN BEGIN
          CARRY   := CARRY   - [CP];
          IF CLASS <> LAKE THEN BEGIN
            PRESENT := PRESENT + [CP];
            IF CP = CLAM THEN
              IF CLASS = BEACH THEN WITH NOUNS[CLAM] DO BEGIN
                CLM := ONECM;
                IF CLAMOPN = NEVER THEN BEGIN
                  CLAMOPN := HASBEEN;
                  PRESENT := PRESENT + [PEARL];
                  CLM := TWOCM; END; END;
            IF CP = VASE THEN
              IF NOT (PILLOW IN PRESENT) THEN BEGIN
                PRESENT := PRESENT - [VASE];
                PRESENT := PRESENT + [SHARD];
                WRITELN('THE MING VASE DELICATELY BREAKS.'); END
              ELSE BEGIN
                WRITE  ('THE MING VASE DELICATELY LANDS ');
                WRITELN('ON THE PILLOW.'); END; END; END
        ELSE BEGIN
          WRITELN('YOU''RE NOT CARRYING A ',PWORD,'.');
          GO := FALSE; END;
    END; (* DODROP *)

  BEGIN (* ACTIONDROP *)
    WITH WHERE↑ DO BEGIN
```

```
      CLM := NOCM;
      GO := TRUE;
      IF WORDPTR↑.NOUNVAL = ALL THEN
        IF (CARRY - [BOAT,BEAR]) = [] THEN
          WRITELN('YOU''VE NOTHING TO DROP.')
        ELSE BEGIN
          FOR CP := AXE TO CLAM DO
            IF CP IN (CARRY - [BOAT,BEAR]) THEN
              DODROP; END
      ELSE
        REPEAT
          CP := WORDPTR↑.NOUNVAL;
          DODROP;
          READTOKEN;
        UNTIL (WORDPTR↑.MEANING <> KNOWN) OR (NOT GO);
      IF GO THEN
        IF CLASS = LAKE THEN
          WRITELN('EVERYTHING DROPPED INTO THE LAKE.')
        ELSE
          IF WORDPTR↑.MEANING = KNOWN THEN
            WRITELN('EVERYTHING HAS BEEN DROPPED.')
          ELSE
            WRITELN('OK.');
      IF CLM > NOCM THEN BEGIN
        WRITE  ('THE CLAM TOUCHES THE ');
        WRITELN('WATER AND OPENS MOMENTARILY.');
        IF CLM = TWOCM THEN BEGIN
          WRITE  ('WHILE THE CLAM IS OPEN, ');
          WRITELN('SOMETHING ROLLS OUT.'); END; END; END;
  END; (* ACTIONDROP *)

PROCEDURE ACTIONDESCRIBE;
    VAR CP : NOUN;
        N  : INTEGER;
  BEGIN (* ACTIONDESCRIBE *)
    WITH WHERE↑ DO BEGIN
      CP := NILL;
      IF WORDPTR↑.NOUNVAL = ALL THEN BEGIN
        IF CARD(CARRY+PRESENT) > 1 THEN
          WRITELN('I SEE THE FOLLOWING THINGS TO DESCRIBE:')
        ELSE
          WRITELN('I SEE THE FOLLOWING THING TO DESCRIBE:');
        FOR CP := AXE TO WOLF DO
          IF CP IN (CARRY + PRESENT) THEN
            TELLNOUN(CP,FALSE); END
      ELSE
        REPEAT
          WITH WORDPTR↑ DO BEGIN
            IF MEANING = UNKNOWN THEN BEGIN
              IF CP = NILL THEN
                WRITELN('I DON''T KNOW WHAT TO DESCRIBE.'); END
            ELSE IF MEANING <> KNOWN THEN BEGIN
              IF CP = NILL THEN
                WRITELN('I CAN ONLY DESCRIBE OBJECTS.'); END
            ELSE BEGIN
              CP := NOUNVAL;
              IF CP IN (CARRY + PRESENT) THEN
                TELLNOUN(CP,FALSE)
              ELSE
                WRITELN('I SEE NO ',PWORD,' HERE.'); END; END;
          READTOKEN;
        UNTIL WORDPTR = VOID; END;
  END; (* ACTIONDESCRIBE *)
```

```
PROCEDURE ACTIONINVEN;
    VAR CP    : NOUN;
        TSET : NOUNSET;
  BEGIN (* ACTIONINVEN *)
    TSET := CARRY - [BOAT,BEAR];
    IF TSET = [] THEN
      WRITELN('YOU''RE NOT CARRYING ANYTHING.')
    ELSE BEGIN
      IF CARD(TSET) > 1 THEN
        WRITELN('YOU''RE CARRYING THE FOLLOWING THINGS:')
      ELSE
        WRITELN('YOU''RE CARRYING THE FOLLOWING THING:');
      FOR CP := AXE TO WOLF DO
        IF CP IN TSET THEN
          TELLNOUN(CP,TRUE); END;
  END; (* ACTIONINVEN *)

PROCEDURE ACTIONFILL;
  BEGIN (* ACTIONFILL *)
    READTOKEN;
    WITH WORDPTR↑, WHERE↑, NOUNS[BOTTLE] DO BEGIN
      IF MEANING = KNOWN THEN
        IF NOUNVAL = LAMP THEN BEGIN
          IF LAMP IN CARRY THEN
            IF NOUNS[LAMP].TIMELEFT <= 15 THEN
              IF NOUNS[LAMP].BURNING THEN BEGIN
                WRITE  ('YOU CAN''T FILL YOUR LANTERN ');
                WRITELN('WHILE ITS BURNING.'); END
              ELSE
                IF (BOTTLE IN CARRY) AND
                   (BOTTLECONTENTS = OILINBOTTLE) THEN BEGIN
                  NOUNS[LAMP].TIMELEFT := BURNTIME;
                  BOTTLECONTENTS := EMPTYBOTTLE;
                  WRITELN('OK.'); END
                ELSE IF POOL = OILPOOL THEN BEGIN
                  NOUNS[LAMP].TIMELEFT := BURNTIME;
                  WRITELN('OK.'); END
                ELSE BEGIN
                  WRITE  ('THERE IS NOTHING HERE TO ');
                  WRITELN('FILL YOUR LANTERN.'); END
            ELSE
              WRITELN('IT IS NOT TIME TO FILL THE LANTERN.')
          ELSE
            WRITELN('YOU''RE NOT CARRYING THE LANTERN.'); END
        ELSE IF NOUNVAL = BOTTLE THEN BEGIN
          READTOKEN;
          IF NOT (BOTTLE IN CARRY) THEN
            WRITELN('YOU''RE NOT CARRYING A BOTTLE.')
          ELSE IF BOTTLECONTENTS <> EMPTYBOTTLE THEN
            WRITELN('YOUR BOTTLE IS ALREADY FILLED.')
          ELSE IF WORDPTR = VOID THEN
            IF POOL <> NOPOOL THEN
              CASE POOL OF
                OILPOOL    : BEGIN
                               BOTTLECONTENTS := OILINBOTTLE;
                               WRITE  ('YOUR BOTTLE IS ');
                               WRITELN('NOW FULL OF OIL.'); END;
                WATERPOOL  : BEGIN
                               BOTTLECONTENTS := WATERINBOTTLE;
                               WRITE  ('YOUR BOTTLE IS ');
                               WRITELN('NOW FULL OF WATER.'); END;
              END
            ELSE BEGIN
              WRITE  ('THERE IS NOTHING HERE WITH ');
              WRITELN('WHICH TO FILL YOUR BOTTLE.'); END
          ELSE IF WORDPTR↑.MEANING <> KNOWN THEN
            BLEWIT
          ELSE IF WORDPTR↑.NOUNVAL = OIL THEN
            IF POOL = OILPOOL THEN BEGIN
              BOTTLECONTENTS := OILINBOTTLE;
              WRITELN('YOUR BOTTLE IS NOW FULL OF OIL.'); END
            ELSE
              WRITELN('THERE IS NO OIL HERE.')
          ELSE IF WORDPTR↑.NOUNVAL = WATER THEN
            IF POOL = WATERPOOL THEN BEGIN
              BOTTLECONTENTS := WATERINBOTTLE;
              WRITELN('YOUR BOTTLE IS NOW FULL OF WATER.'); END
            ELSE
              WRITELN('THERE IS NO WATER HERE.')
          ELSE
            BLEWIT; END
        ELSE
          BLEWIT
      ELSE
        BLEWIT; END;
  END; (* ACTIONFILL *)

PROCEDURE ACTIONEMPTY;
    LABEL 77;
  BEGIN (* ACTIONEMPTY *)
    WITH WORDPTR↑ DO
      IF MEANING = KNOWN THEN
        IF NOT (NOUNVAL IN [BOTTLE,OIL,WATER]) THEN
          BLEWIT
        ELSE IF BOTTLE IN CARRY THEN WITH NOUNS[BOTTLE] DO
          CASE NOUNVAL OF
            BOTTLE : ;
            OIL    : IF BOTTLECONTENTS <> OILINBOTTLE THEN BEGIN
                       WRITE  ('YOU''RE BOTTLE DOES ');
                       WRITELN('NOT CONTAIN OIL.');
                       GOTO 77; END;
            WATER  : IF BOTTLECONTENTS <> WATERINBOTTLE THEN BEGIN
                       WRITE  ('YOU''RE BOTTLE DOES ');
                       WRITELN('NOT CONTAIN WATER.');
                       GOTO 77; END;
          END;
    WITH WHERE↑, NOUNS[BOTTLE] DO
      IF NOT (BOTTLE IN CARRY) THEN BEGIN
        WRITE  ('YOU''RE NOT CARRYING ANYTHING ');
        WRITELN('THAT CAN BE EMPTIED.'); END
      ELSE
        CASE BOTTLECONTENTS OF
          EMPTYBOTTLE   : WRITELN('YOUR BOTTLE IS EMPTY.');
          OILINBOTTLE   : BEGIN
                            BOTTLECONTENTS := EMPTYBOTTLE;
                            IF CLASS <> LAKE THEN
                              IF (WARN = 3) AND BLOCK THEN BEGIN
                                REMOVEBLOCK;
                                WRITE  ('THE OIL FREES THE DOOR ');
                                WRITELN('AND IT SWINGS OPEN.'); END
                              ELSE BEGIN
                                WET := OILWET;
                                WRITE  ('THE GROUND IS NOW ');
                                WRITELN('WET WITH OIL.'); END
                            ELSE BEGIN
                              WRITE  ('YOU HAVE POLLUTED THE ');
                              WRITELN('LAKE WITH  OIL.'); END; END;
          WATERINBOTTLE : BEGIN
```

```
                                BOTTLECONTENTS := EMPTYBOTTLE;
                                IF CLASS <> LAKE THEN BEGIN
                                  IF WARN = 4 THEN
                                    IF NOUNS[PLANT].HEIGHT < OVERGROWN
                                       THEN BEGIN
                                      NOUNS[PLANT].HEIGHT :=
                                        SUCC(NOUNS[PLANT].HEIGHT);
                                      TELLNOUN(PLANT,FALSE); END
                                    ELSE BEGIN
                                      WRITE  ('WATERING THE PLANT ');
                                      WRITELN('IS NOW USELESS.');
                                      WET := WATERWET;
                                      WRITE  ('THE GROUND IS NOW ');
                                      WRITELN('WET WITH WATER.'); END
                                  ELSE BEGIN
                                    WET := WATERWET;
                                    WRITE  ('THE GROUND IS NOW WET ');
                                    WRITELN('WITH WATER.'); END; END
                                ELSE BEGIN
                                  WRITE  ('YOU''VE EMPTIED YOUR ');
                                  WRITELN('BOTTLE IN THE LAKE.');
                                  END; END;
           END;



7:   END; (* ACTIONEMPTY *)

   PROCEDURE ACTIONON;
     BEGIN (* ACTIONON *)
       WITH WORDPTR↑, WHERE↑, NOUNS[LAMP] DO BEGIN
         IF MEANING = KNOWN THEN
           IF NOUNVAL <> LAMP THEN
             BLEWIT;
         IF NOT (LAMP IN (CARRY + PRESENT)) THEN
           IF LIGHT THEN
             WRITELN('I SEE NO LAMP HERE.')
           ELSE
             WRITELN('IT''S TOO DARK TO SEE ANYTHING.')
         ELSE IF BURNING THEN
           WRITELN('THE LANTERN IS ALREADY ON.')
         ELSE IF NOT (MATCH IN (CARRY + PRESENT)) THEN
           WRITELN('YOU NEED MATCHES TO LIGHT THE LANTERN.')
         ELSE IF NOUNS[MATCH].NOFMATCHES <= 0 THEN
           WRITELN('SORRY, YOU''RE OUT OF MATCHES.')
         ELSE BEGIN
           NOUNS[MATCH].NOFMATCHES := NOUNS[MATCH].NOFMATCHES - 1;
           IF NOUNS[MATCH].NOFMATCHES <= 0 THEN
             WRITELN('THAT WAS YOUR LAST MATCH.');
           IF TIMELEFT <= 0 THEN
             WRITELN('THE LANTERN HAS NO MORE FUEL IN IT.')
           ELSE BEGIN
             BURNING := TRUE;
             WRITELN('THE LANTERN IS ON.');
             IF SIDE = INSIDE THEN
               TELLLOCATION(BRIEFLY); END; END; END;
     END; (* ACTIONON *)

   PROCEDURE ACTIONOFF;
     BEGIN (* ACTIONOFF *)
       WITH WORDPTR↑ DO
         IF MEANING = KNOWN THEN
           IF NOUNVAL <> LAMP THEN
```

```
          BLEWIT;
      IF LAMP IN (CARRY + WHERE↑.PRESENT) THEN WITH NOUNS[LAMP] DO
        IF BURNING THEN BEGIN
          BURNING := FALSE;
          WRITELN('THE LANTERN IS OFF.'); END
        ELSE
          WRITELN('THE LANTERN IS ALREADY OFF.')
      ELSE
        IF LIGHT THEN
          WRITELN('I SEE NO LANTERN HERE.')
        ELSE
          WRITELN('IT''S TOO DARK TO SEE ANYTHING.');
    END; (* ACTIONOFF *)

  PROCEDURE ACTIONDRINK;
    BEGIN (* ACTIONDRINK *)
      WITH WORDPTR↑, WHERE↑, NOUNS[BOTTLE] DO
        IF (MEANING <> KNOWN) OR
           ((MEANING = KNOWN) AND (NOUNVAL = BOTTLE)) THEN
          IF BOTTLE IN CARRY THEN
            CASE BOTTLECONTENTS OF
              EMPTYBOTTLE    : WRITELN('YOUR BOTTLE IS EMPTY.');
              OILINBOTTLE    : WRITELN('UGH.  OIL IS NOT PALATABLE.');
              WATERINBOTTLE  : BEGIN
                                 WRITELN('THANK YOU, I WAS THIRSTY.');
                                 WRITELN('YOUR BOTTLE IS NOW EMPTY.');
                                 BOTTLECONTENTS := EMPTYBOTTLE; END;
            END
          ELSE
            WRITELN('YOU''RE NOT CARRYING A BOTTLE.')
        ELSE IF NOUNVAL = OIL THEN
          IF POOL = OILPOOL THEN
            WRITELN('UGH.  OIL IS NOT PALATABLE.')
          ELSE
            WRITELN('I SEE NO OIL HERE.')
        ELSE IF NOUNVAL = WATER THEN
          IF POOL = WATERPOOL THEN
            WRITELN('THANK YOU, I WAS THIRSTY.')
          ELSE
            WRITELN('I SEE NO WATER HERE.')
        ELSE
          BLEWIT;
    END; (* ACTIONDRINK *)

  PROCEDURE ACTIONEAT;
    BEGIN (* ACTIONEAT *)
      WITH WORDPTR↑ DO
        IF MEANING = KNOWN THEN
          IF NOUNVAL <> FOOD THEN
            BLEWIT;
      IF FOOD IN CARRY THEN BEGIN
        CARRY := CARRY - [FOOD];
        WRITELN('THANK YOU.  THE FOOD IS QUITE TASTY.'); END
      ELSE
        WRITELN('I SEE NOTHING TO EAT IN YOUR POSSESSION.');
    END; (* ACTIONEAT *)

  PROCEDURE ACTIONSWIM;
    BEGIN (* ACTIONSWIM *)
      IF WHERE↑.CLASS IN [BEACH,LAKE] THEN BEGIN
        WRITELN('BLUB!  BLUB!  I FORGOT TO TELL YOU THAT FULLY');
        WRITELN('EQUIPED ADVENTURERS ARE TOO CLUMSY TO SWIM.');
        OOPS; END
      ELSE
      WRITELN('I DON''T SEE ENOUGH WATER TO SWIM IN AROUND HERE.');
  END; (* ACTIONSWIM *)

PROCEDURE ACTIONUNLOCK;
  BEGIN (* ACTIONUNLOCK *)
    WITH WHERE↑ DO
      IF NOT (KEYS IN CARRY) THEN
        WRITELN('YOU''RE NOT CARRYING ANY KEYS.')
      ELSE IF WARN IN [ 1 .. 2 ] THEN
        IF BLOCK THEN BEGIN
          REMOVEBLOCK;
          WARNING; END
        ELSE
          WRITELN('THE LOCK IS ALREADY UNLOCKED.')
      ELSE IF CHEST IN PRESENT THEN
        IF ROD IN CARRY THEN WITH NOUNS[CHEST] DO
          IF CHESTSTATE = LOCKEDCHEST THEN BEGIN
            CHESTSTATE := TREASURECHEST;
            TOLD := FALSE;
            WRITE  ('THE CHEST OPENS REVEALING ');
            WRITELN('THE TREASURE INSIDE.'); END
          ELSE
            WRITELN('THE CHEST IS ALREADY OPEN.')
        ELSE BEGIN
          WRITE  ('YOU DON''T HAVE EVERYTHING ');
          WRITELN('YOU NEED TO OPEN THE CHEST.'); END
      ELSE
        WRITELN('THERE IS NOTHING HERE TO UNLOCK.');
  END; (* ACTIONUNLOCK *)

PROCEDURE ACTIONLOCK;
  BEGIN (* ACTIONLOCK *)
    WITH WHERE↑ DO
      IF NOT (KEYS IN CARRY) THEN
        WRITELN('YOU''RE NOT CARRYING ANY KEYS.')
      ELSE IF WARN IN [ 1 .. 2 ] THEN
        IF BLOCK THEN
          WRITELN('THE LOCK IS ALREADY LOCKED.')
        ELSE BEGIN
          SETBLOCK;
          WARNING; END
      ELSE
        WRITELN('I SEE NO LOCK AROUND HERE.');
  END; (* ACTIONLOCK *)

PROCEDURE ACTIONROW;
  BEGIN (* ACTIONROW *)
    READTOKEN;
    IF WORDPTR↑.MEANING = DIRECT THEN BEGIN
      ROWFLAG := TRUE;
      DOMOVEMENT;
      ROWFLAG := FALSE; END
    ELSE
      WRITELN('WHICH DIRECTION DO YOU WANT TO ROW.');
  END; (* ACTIONROW *)

PROCEDURE ACTIONRUB;
  BEGIN
    READTOKEN;
    WITH WORDPTR↑ DO
      IF NOUNVAL IN CARRY THEN
        IF NOUNVAL = ROD THEN
         WRITELN('RUBBING THE ROD DOESN''T DO ANYTHING HERE')
        ELSE
```

```
          IF NOUNVAL = LAMP THEN
            WRITELN('SHINING THE LAMP DOESN''T DO ANYTHING INTERESTING')
          ELSE
            WRITELN('NOTHING INTERESTING HAPPENS ')
       ELSE
         WRITELN('YOU DON''T HAVE IT');
   END; (* ACTIONRUB *)

 PROCEDURE SUPPLYHELP;
   BEGIN (* SUPPLYHELP *)
     WRITELN('  I KNOW OF DIRECTIONS, ACTIONS, AND OBJECTS.  TO  ');
     WRITELN('MOVE FROM ONE PLACE TO ANOTHER, USE COMPASS POINTS ');
     WRITELN('OR DIRECTIONS LIKE:  EAST, DOWN, OR ENTER.  RARELY,');
     WRITELN('A MAGIC WORD WILL MOVE YOU FROM ONE PLACE TO       ');
     WRITELN('ANOTHER.  IF YOU KNOW THE EXACT ROUTE, YOU MAY LIST');
     WRITELN('A SERIES OF DIRECTIONS AND I WILL FOLLOW THEM.     ');
     WRITELN('  I KNOW ABOUT MANY OBJECTS.  TO MANIPULATE        ');
     WRITELN('OBJECTS, USE SOME ACTION WORD FOLLOWED BY AN       ');
     WRITELN('OBJECT.  TO PICK UP A ROD, SAY ''TAKE ROD''.       ');
     WRITELN('SOMETIMES, IF YOU OMIT THE OBJECT, I WILL ASSUME   ');
     WRITELN('ALL OBJECTS PRESENT.  OBJECTS CAN HAVE SIDE        ');
     WRITELN('EFFECTS.  THE ROD SCARES THE BIRD.  SOME OBJECTS   ');
     WRITELN('WILL CHANGE THE CAVERN IF PROPERLY USED.           ');
     WRITELN('  SOME HELPFUL WORDS ARE: ''LOOK'' - LOOK AROUND AT');
     WRITELN('YOUR PRESENT POSITION; ''DESCRIBE OBJECT'' - I WILL');
     WRITELN('TELL YOU MORE ABOUT AN OBJECT; AND ''INVENTORY'' - I');
     WRITELN('WILL LIST WHAT YOU''RE CARRYING.                   ');
     WRITELN('  USUALLY, PEOPLE HAVING TROUBLE ARE TRYING        ');
     WRITELN('SOMETHING BEYOND MY CAPABILITIES AND SHOULD TRY A  ');
     WRITELN('COMPLETELY DIFFERENT TACK.  ALSO, CAVE PASSAGES    ');
     WRITELN('TURN A LOT, AND LEAVING A ROOM TO THE NORTH DOESN''T');
     WRITELN('GUARANTEE YOU CAN GO BACK BY GOING SOUTH.          ');
     WRITELN('  GOOD LUCK!                                       ');
   END; (* SUPPLYHELP *)

 PROCEDURE SUPPLYINFO;
   BEGIN (* SUPPLYINFO *)
     WRITELN('  IF YOU WANT TO END YOUR ADVENTURE EARLY, SAY     ');
     WRITELN('''QUIT''.  IF YOU GET INTO TROUBLE AND CAN''T FIND A');
     WRITELN('WAY OUT, SAY ''RESIGN''.  TO SEE HOW WELL YOU''RE  ');
     WRITELN('DOING, SAY ''SCORE''.  TO SAVE PAPER, SAY ''BRIEF'',');
     WRITELN('AND I''LL TELL YOU THE FULL DESCRIPTION OF A ROOM  ');
     WRITELN('ONLY THE FIRST TIME YOU GET THERE.  TO ALWAYS GET  ');
     WRITELN('THE FULL DESCRIPTION, SAY ''VERBOSE''.  TO SUSPEND ');
     WRITELN('YOUR ADVENTURE, SAY ''SAVE NAME''.                 ');
     WRITELN('  TO GET FULL CREDIT FOR A TREASURE, YOU MUST HAVE ');
     WRITELN('LEFT IT SAFELY IN THE WELLHOUSE.  YOU DO GET POINTS');
     WRITELN('FOR JUST DISCOVERING TREASURES AND EXPLORING THE   ');
     WRITELN('CAVERN.  YOU LOSE POINTS FOR GETTING KILLED OR     ');
     WRITELN('RESIGNING.                                         ');
   END; (* SUPPLYINFO *)

 PROCEDURE ACTIONBUILD;
     VAR DI : DIRECTION;
   BEGIN (* ACTIONBUILD *)
     WITH WHERE↑, WORDPTR↑ DO BEGIN
       IF NOT (NOUNVAL IN [BRIDGE, LADDER]) THEN
         BLEWIT
       ELSE IF NOT ([HAMMER,NAIL, WOOD] <=
                   (CARRY + PRESENT)) THEN BEGIN
         WRITE  ('YOU DON''T HAVE ALL THE THINGS ');
         WRITELN('YOU NEED TO BUILD A ',PWORD,'.'); END
       ELSE BEGIN
         WITH NOUNS[NAIL] DO BEGIN
```

```
          NOFNAILS := NOFNAILS - 1;
          IF NOFNAILS <= 0 THEN BEGIN
            WRITELN('YOU''VE USED YOUR LAST NAILS.');
            CARRY := CARRY - [NAIL];
            PRESENT := PRESENT - [NAIL]; END; END;
        WITH NOUNS[WOOD] DO BEGIN
          IF NOUNVAL = LADDER THEN
            PILESIZE := PILESIZE - 1
          ELSE
            PILESIZE := PILESIZE - 3;
          IF PILESIZE <= 0 THEN BEGIN
            WRITELN('YOU''VE USED YOUR LAST WOOD.');
            CARRY := CARRY - [WOOD];
            PRESENT := PRESENT - [WOOD]; END; END;
        PRESENT := PRESENT + [NOUNVAL];
        WRITELN('YOU''VE BUILT A ',PWORD,'.');
        IF (WARN = 9) AND (NOUNVAL = BRIDGE) THEN
          REMOVEBLOCK; END; END;
  END; (* ACTIONBUILD *)

PROCEDURE ACTIONRAISE;
  BEGIN (* ACTIONRAISE *)
    WITH WHERE↑ DO BEGIN
      IF WORDPTR↑.NOUNVAL <> LADDER THEN
        BLEWIT
      ELSE IF NOT (LADDER IN (CARRY + PRESENT)) THEN
        WRITELN('THERE IS NO LADDER HERE TO RAISE.')
      ELSE BEGIN
        CARRY := CARRY - [LADDER];
        PRESENT := PRESENT + [LADDER];
        IF WARN = 8 THEN BEGIN
          REMOVEBLOCK;
          WARNING; END
        ELSE
          WRITELN('THE LADDER HAS BEEN RAISED.'); END; END;
  END; (* ACTIONRAISE *)

PROCEDURE ACTIONTHROW;
  BEGIN (* ACTIONTHROW *)
    WITH WORDPTR↑, WHERE↑ DO
      IF NOUNVAL = ROD THEN
        IF ROD IN CARRY THEN BEGIN
          CARRY   := CARRY   - [ROD];
          PRESENT := PRESENT + [ROD];
          IF WARN = 5 THEN BEGIN
            CHANGEBLOCK;
            WARNING; END
          ELSE
            IF CLASS = LAND THEN
              WRITELN('NOTHING UNUSUAL HAPPENS.')
            ELSE BEGIN
              PRESENT := PRESENT - [ROD];
              WRITELN('THE ROD FALLS INTO THE LAKE.'); END; END
        ELSE
          WRITELN('YOU''RE NOT CARRYING A ROD.')
      ELSE IF NOUNVAL = ROPE THEN
        IF ROPE IN CARRY THEN BEGIN
          CARRY   := CARRY   - [ROPE];
          PRESENT := PRESENT + [ROPE];
          IF WARN = 7 THEN BEGIN
            REMOVEBLOCK;
            WARNING; END
          ELSE
            IF CLASS = LAND THEN
          WRITELN('THE ROPE FALLS TO THE GROUND.')
        ELSE BEGIN
          PRESENT := PRESENT - [ROPE];
          WRITELN('THE ROPE FALLS INTO THE LAKE.'); END; END
    ELSE
      WRITELN('YOU''RE NOT CARRYING A ROPE.')
  ELSE IF NOUNVAL = BIRD THEN
    IF BIRD IN CARRY THEN BEGIN
      CARRY   := CARRY   - [BIRD];
      PRESENT := PRESENT + [BIRD];
      NOUNS[CAGE].CAGECONTENTS := EMPTYCAGE;
      NOUNS[BIRD].BIRDSTATE    := FREEBIRD;
      IF SNAKE IN PRESENT THEN BEGIN
        PRESENT := PRESENT - [SNAKE];
        WRITE  ('THE BIRD ATTACKS THE SNAKE ');
        WRITELN('AND IN AN ASTONISHING FLURRY');
        WRITELN('THE SNAKE IS DRIVEN AWAY.');
        STRANGLE := 0; END
      ELSE IF DRAGON IN PRESENT THEN BEGIN
        PRESENT := PRESENT - [BIRD];
        WRITE  ('THE BIRD ATTACKS THE DRAGON ');
        WRITELN('AND IN AN ASTONISHING FLURRY');
        WRITELN('THE DRAGON BURNS THE BIRD TO A CINDER.'); END
      ELSE
        IF CLASS = LAND THEN BEGIN
          WRITE  ('THE BIRD FLUTTERS IN THE ');
          WRITELN('AIR AND LANDS NEARBY.'); END
        ELSE BEGIN
          PRESENT := PRESENT - [BIRD];
          WRITE  ('THE BIRD FLUTTERS IN THE ');
          WRITELN('AIR AND FLIES AWAY.'); END; END
    ELSE
      WRITELN('YOU''RE NOT CARRYING A BIRD.')
  ELSE IF NOUNVAL IN [ AXE, KNIFE ] THEN
    IF NOUNVAL IN CARRY THEN BEGIN
      CARRY := CARRY - [NOUNVAL];
      IF CLASS = LAND THEN
        PRESENT := PRESENT + [NOUNVAL];
      IF ORC IN PRESENT THEN
        IF RANDOM(0) < 0.5 THEN BEGIN
          WRITELN('YOU''VE KILLED A ORC.');
          WRITE  ('HE DISAPPEARS IN A CLOUD ');
          WRITELN('OF GREASY BLACK SMOKE.');
          PRESENT := PRESENT - [ORC]; END
        ELSE BEGIN
          WRITE  ('THE ',PWORD,' BOUNCES HARMLESSLY ');
          WRITELN('OFF THE ORC.'); END
      ELSE IF DRAGON IN PRESENT THEN
        IF NOUNVAL = AXE THEN
          IF RANDOM(0) < 0.33 THEN BEGIN
            WRITELN('YOU''VE KILLED THE DRAGON.');
            WRITE  ('IT CONTRACTS INTO ');
            WRITELN('WRINKLES AND DISAPPEARS.');
            PRESENT := PRESENT - [DRAGON];
            STRANGLE := 0;
            RIGHTECH := FALSE; END
          ELSE BEGIN
            RIGHTECH := TRUE;
            WRITE  ('THE AXE BOUNCES HARMLESSLY ');
            WRITELN('OFF THE DRAGON.'); END
        ELSE BEGIN
          WRITE  ('THE KNIFE BOUNCES HARMLESSLY ');
          WRITELN('OFF THE DRAGON.'); END
      ELSE
```

```
            IF CLASS = LAND THEN
              IF ([BEAR,CLAM,SNAKE,TROLL,WOLF]*PRESENT)
                  <> [] THEN BEGIN
                IF BEAR IN PRESENT THEN BEGIN
                  WRITE  ('THE ',PWORD,' BOUNCES ');
                  WRITELN('HARMLESSLY OFF THE BEAR.'); END
                ELSE IF CLAM IN PRESENT THEN BEGIN
                  WRITE  ('THE ',PWORD,' BOUNCES ');
                  WRITELN('HARMLESSLY OFF THE CLAM.'); END
                ELSE IF SNAKE IN PRESENT THEN BEGIN
                  WRITE  ('THE ',PWORD,' BOUNCES HARMLESSLY ');
                  WRITELN('OFF THE SNAKE.'); END
                ELSE IF TROLL IN PRESENT THEN BEGIN
                  WRITE  ('THE ',PWORD,' BOUNCES HARMLESSLY');
                  WRITELN(' OFF THE TROLL.'); END
                ELSE IF WOLF IN PRESENT THEN BEGIN
                  IF NOUNS[WOLF].LIFE = DEAD THEN BEGIN
                     WRITELN('AWW, LEAVE THE POOR WOLF ALONE');
                      CARRY := CARRY + [NOUNVAL];
                     PRESENT := PRESENT - [NOUNVAL] END
                  ELSE
                  IF RANDOM(0) < 0.45 THEN
                    BEGIN
                    WRITELN(' YOU KILLED THE WOLF ');
                    NOUNS[WOLF].LIFE := DEAD;
                    END
                  ELSE
                    BEGIN
                     WRITE  ('THE ',PWORD,' BOUNCES HARMLESSLY ');
                     WRITELN('OFF THE WOLF.'); END; END; END
              ELSE IF BIRD IN PRESENT THEN BEGIN
                PRESENT := PRESENT - [BIRD];
                BOATLOC↑.PRESENT := BOATLOC↑.PRESENT + [BIRD];
                WRITE  ('THE ',PWORD,' MISSES THE BIRD ');
                WRITELN('AND HE FLIES AWAY.'); END
              ELSE
                WRITELN('THE ',PWORD,' FALLS TO THE GROUND.')
            ELSE
              WRITELN('THE ',PWORD,' FALLS INTO THE WATER.'); END
        ELSE
          IF NOUNVAL = AXE THEN
            WRITELN('YOU''RE NOT CARRYING AN AXE.')
          ELSE
            WRITELN('YOU''RE NOT CARRYING A KNIFE.')
      ELSE
        BLEWIT;
  END; (* ACTIONTHROW *)

PROCEDURE ACTIONWAVE;
    VAR TEMP : LOCALE;
        ENEMY : NOUN;
  BEGIN (* ACTIONWAVE *)
    WITH WORDPTR↑, WHERE↑ DO
      IF NOUNVAL = AXE THEN
        IF AXE IN CARRY THEN
          BEGIN
          IF ([DRAGON,SNAKE,ORC,BEAR,TROLL,WOLF] * PRESENT) <>[]
          THEN
            BEGIN
            IF DRAGON IN PRESENT THEN ENEMY:=DRAGON ELSE
            IF ORC IN PRESENT THEN ENEMY := ORC ELSE
            IF SNAKE IN PRESENT THEN ENEMY := SNAKE ELSE
            IF TROLL IN PRESENT THEN ENEMY := TROLL ELSE
            IF BEAR IN PRESENT THEN ENEMY := BEAR ELSE
```

```
             IF (WOLF IN PRESENT) AND (NOUNS[WOLF].LIFE = ALIVE) THEN
                ENEMY := WOLF;
             IF RANDOM(0)<0.50 THEN
                BEGIN
                WRITELN('YOU KILLED A '    );
                TELLNOUN(ENEMY,TRUE);
                IF ENEMY <> WOLF THEN
                  WRITELN('IT SHRIVELS UP AND DISAPEARS')
                ELSE NOUNS[WOLF].LIFE := DEAD;
                END
             ELSE
               BEGIN
               RIGHTECH := TRUE;
               WRITELN('A MIGHTY SWING, BUT IT MISSED');
               END
             END (* AN ENEMY *)
           ELSE
             WRITELN('THERE ARE NO ENEMIES HERE')
           END (* AXE IN CARRY *)
         ELSE
           WRITELN('YOU''RE NOT CARRYING AN AXE')
       ELSE (* NOT AXE *)
       IF NOUNVAL = ROD THEN
         IF ROD IN CARRY THEN
           IF WARN = 6 THEN BEGIN
             CHANGEBLOCK;
             WARNING; END
           ELSE
             CASE TRUNC ( 2.999 * RANDOM(0) ) OF
               0: WRITELN('NOTHING UNUSUAL HAPPENS.');
               1: WRITELN('NOTHING PECULIAR HAPPENS.');
               2: IF LIGHT THEN
                    WRITELN('STRANGE, THAT SOUNDED LIKE THUNDER.')
                  ELSE BEGIN
                    WRITELN('A LIGHTNING BOLT FLASHES OVERHEAD.');
                    WRITELN('FOR A MOMENT YOU SEE:');
                    TEMP := SIDE;
                    SIDE := OUTSIDE;
                    TELLLOCATION(FALSE);
                    SIDE := TEMP; END;
             END
         ELSE
           WRITELN('YOU''RE NOT CARRYING A ROD.')
       ELSE
         BLEWIT;
  END; (* ACTIONWAVE *)

PROCEDURE ACTIONFEED;
    VAR CP : NOUN;
  BEGIN (* ACTIONFEED *)
    WITH WORDPTR↑, WHERE↑ DO
      IF NOT (FOOD IN CARRY) THEN
        WRITELN('YOU''RE NOT CARRYING ANY FOOD.')
      ELSE BEGIN
        IF WORDPTR <> VOID THEN
          IF NOT (NOUNVAL IN [ BEAR .. WOLF ]) THEN
            BLEWIT
          ELSE
            CP := NOUNVAL
        ELSE BEGIN
          CP := BEAR;
          WHILE NOT (CP IN (CARRY + PRESENT)) AND (CP < WOLF) DO
            CP := SUCC(CP);
          IF NOT (CP IN (CARRY + PRESENT)) THEN
          BLEWIT; END;
      IF CP IN [ BEAR, DRAGON, SNAKE, WOLF ] THEN
        CARRY := CARRY - [FOOD];
      IF CP IN [ BEAR, DRAGON, SNAKE, WOLF, BIRD, CLAM, ORC,
        PIRATE, TROLL ] THEN BEGIN
        CASE CP OF
          BEAR   : BEGIN
                     NOUNS[BEAR].BEARSTATE := HAPPY;
                     WRITE  ('THE BEAR EATS YOU''RE FOOD AND ');
                     WRITELN('BECOMES RATHER FRIENDLY.'); END;
          BIRD   :  BEGIN
                      WRITE  ('THE BIRD NEEDS SEED, ');
                      WRITELN('NOT YOUR SANDWICH.'); END;
          CLAM   : BEGIN
                     WRITE  ('YOU CAN''T FEED A CLAM FOOD, ');
                     WRITELN('IT NEEDS A BODY OF WATER.'); END;
          DRAGON : BEGIN
                     WRITE  ('THE DRAGON EATS THE FOOD AND ');
                     WRITELN('EYES YOU HUNGRILY.'); END;
          ORC   : BEGIN
                     WRITE  ('THE ORC IS TO ANGRY ');
                     WRITELN('TO EAT FOOD.'); END;
          PIRATE : BLEWIT;
          SNAKE  : BEGIN
                     WRITE  ('THE SNAKE EATS THE FOOD ');
                     WRITELN('AND EYES YOU HUNGRILY.'); END;
          TROLL  : BEGIN
                     WRITE  ('THE GIANT SNARLS AND DEMANDS ');
                     WRITELN('A TREASURE, NOT FOOD.'); END;
          WOLF   : BEGIN
                     WRITE  ('THE WOLF EATS THE FOOD ');
                     WRITELN('AND EYES YOU HUNGRILY.'); END;
        END; END
      ELSE
        WRITELN('THERE''S NOTHING HERE TO FEED.'); END;
  END; (* ACTIONFEED *)

PROCEDURE ACTIONRESIGN;
  BEGIN (* ACTIONRESIGN *)
    WRITELN('I SEE YOU''VE RESIGNED BY HOLDING YOUR BREATH.');
    WRITELN('BY THE WAY, BLUE IS A BAD SKIN COLOR.');
    OOPS;
  END; (* ACTIONRESIGN *)

PROCEDURE ACTIONQUIT;
  BEGIN (* ACTIONQUIT *)
    WRITELN('DO YOU REALLY WANT TO QUIT NOW?');
    QUESTION := QUITQUEST;
  END; (* ACTIONQUIT *)

PROCEDURE ACTIONSAVE;
  BEGIN (* ACTIONSAVE *)
    DEFINITION := TRUE;
    READTOKEN;
    DEFINITION := FALSE;
    IF STRING[1] <> CHR(0) THEN BEGIN
      WRITELN('YOUR ADVENTURE HAS BEEN SAVED ON ',PWORD,'.');
      WRITELN('TO RESUME TYPE:  ''-ADVENT(ADVORG=',PWORD,')''.');
*         SNAP(STRING);    *)
      WRITELN('YOUR ADVENTURE HAS BEEN RESTORED.'); END
    ELSE
      BLEWIT;
  END; (* ACTIONSAVE *)
```

```
BEGIN (* DOACTION *)
  AP := WORDPTR↑.ACTVAL;
  IF NOT (AP IN [FILL, ROW, SAVE, SWIM]) THEN BEGIN
    READTOKEN;
    IF AP IN [BRIEF, HELP..INVEN, LEFT..NO, QUIT, RIGHT,
              SCORE, UNLOCK, VERBOSE, YES] THEN BEGIN
      IF WORDPTR <> VOID THEN
        BLEWIT; END
    ELSE IF AP IN [BUILD, DRINK, EAT..FILL, KILL, OFF, ON,
                   RAISE, THROW, WAVE] THEN BEGIN
      IF WORDPTR↑.MEANING <> KNOWN THEN
        IF AP IN [BUILD, FEED, FILL, KILL, RAISE, THROW,
                  WAVE] THEN
          BLEWIT
        ELSE IF WORDPTR <> VOID THEN
          BLEWIT;
      IF WORDPTR <> VOID THEN BEGIN
        WP := WORDPTR;
        ST := STRING;
        READTOKEN;
        IF WORDPTR <> VOID THEN
          BLEWIT;
        WORDPTR := WP;
        STRING := ST; END; END
    ELSE
      IF WORDPTR = VOID THEN
        WORDPTR := WALL
      ELSE IF WORDPTR↑.MEANING <> KNOWN THEN
        BLEWIT; END;
  IF NOT LIGHT THEN
    IF AP IN [ BUILD .. DRINK, EAT, FEED, INVEN,
              KILL, LOCK, LOOK, OFF, RAISE, SWIM,
              THROW, UNLOCK ] THEN BEGIN
      WRITELN('IT''S TOO DARK TO SEE ANYTHING.');
      GOTO 50; END;
  CASE AP OF
    BRIEF    : BEGIN
                 BRIEFLY := TRUE;
                 WRITELN('OK.  I''LL DESCRIBE LOCATIONS BRIEFLY.');
               END;
    BUILD    : ACTIONBUILD;
    DESCRIBE : ACTIONDESCRIBE;
    DRINK    : ACTIONDRINK;
    DROP     : ACTIONDROP;
    EAT      : ACTIONEAT;
    EMPTY    : ACTIONEMPTY;
    FEED     : ACTIONFEED;
    FILL     : ACTIONFILL;
    HELP     : SUPPLYHELP;
    INFO     : SUPPLYINFO;
    INVEN    : ACTIONINVEN;
    KILL     : BEGIN
                 WRITE  ('PLEASE BE MORE SPECIFIC ');
                 WRITELN('ON HOW TO DO THAT.');
               END;
    LEFT     : BEGIN
                 WRITE  ('I DON''T KNOW HOW TO GO LEFT.  ');
                 WRITELN('USE COMPASS POINTS.');
               END;
    LOCK     : ACTIONLOCK;
    LOOK     : TELLLOCATION(FALSE);
    NO       : BLEWIT;
    OFF      : ACTIONOFF;
    ON       : ACTIONON;
```

```
      QUIT     : ACTIONQUIT;
      RAISE    : ACTIONRAISE;
      RESIGN   : ACTIONRESIGN;
      RIGHT    : BEGIN
                   WRITE  ('I DON''T KNOW HOW TO GO RIGHT.  ');
                   WRITELN('USE COMPASS POINTS.');
                 END;
      ROW      : ACTIONROW;
      RUB      : ACTIONRUB;
      SAVE     : ACTIONSAVE;
      SCORE    : SCOREGAME;
      SWIM     : ACTIONSWIM;
      TAKE     : ACTIONTAKE;
      THROW    : ACTIONTHROW;
      UNLOCK   : ACTIONUNLOCK;
      VERBOSE  : BEGIN
                   BRIEFLY := FALSE;
                   WRITELN('OK.  I''LL DESCRIBE LOCATIONS FULLY.');
                 END;
      WAVE     : ACTIONWAVE;
      YES      : BLEWIT;
    END;
  END; (* DOACTION *)

PROCEDURE ANALYZEQUESTION;

  PROCEDURE RESTART;
    BEGIN (* RESTART *)
      WRITELN;
      WITH STARTLOC↑ DO BEGIN
        IF LAMP IN CARRY THEN BEGIN
          CARRY := CARRY - [LAMP];
          PRESENT := PRESENT + [LAMP]; END;
        IF MATCH IN CARRY THEN BEGIN
          CARRY := CARRY - [MATCH];
          PRESENT := PRESENT + [MATCH];
          NOUNS[LAMP].BURNING := FALSE; END; END;
      IF BOAT IN CARRY THEN
        CARRY := CARRY - [BOAT]
      ELSE
        BOATPOS↑.PRESENT := BOATPOS↑.PRESENT - [BOAT];
      BOATLOC↑.PRESENT := BOATLOC↑.PRESENT + [BOAT];
      WITH WHERE↑ DO
        IF CLASS <> LAKE THEN
          PRESENT := PRESENT + CARRY;
      CARRY := [];
      DONE     := FALSE;
      QUESTION := NOQUEST;
      WHERE    := STARTLOC;
      WAS      := NIL;
      ROWFLAG := FALSE;
    END; (* RESTART *)

  PROCEDURE YNINFO;
    BEGIN (* YNINFO *)
      WITH WORDPTR↑ DO
        IF MEANING = ACT THEN
          IF ACTVAL = NO THEN BEGIN
            WRITELN('VERY WELL.');
            WRITELN;
            QUESTION := NOQUEST; END
          ELSE IF ACTVAL = YES THEN BEGIN
            WRITELN('  WELCOME TO ADVENTURE.  SOMEWHERE NEARBY IS ');
            WRITELN('COLOSSAL CAVE, WHERE OTHERS HAVE FOUND       ');
          WRITELN('FORTUNES IN TREASURE AND GOLD.  THOUGH IT IS ');
          WRITELN('RUMORED THAT SOME WHO ENTER ARE NEVER SEEN   ');
          WRITELN('AGAIN.  MAGIC IS SAID TO WORK IN THE CAVE.  I');
          WRITELN('WILL BE YOUR EYES AND HANDS.  DIRECT ME WITH ');
          WRITELN('COMMANDS OF ONE OR TWO WORDS.  SHOULD YOU GET');
          WRITELN('STUCK, TYPE ''HELP'' FOR SOME GENERAL HINTS. ');
          WRITELN('FOR INFORMATION ON HOW TO END YOUR ADVENTURE ');
          WRITELN('AND OTHER PERTINENT ITEMS, TYPE ''INFO''.    ');
          WRITELN;
          QUESTION := NOQUEST; END
        ELSE
          WRITELN('PLEASE ANSWER THE QUESTION WITH YES OR NO.')
      ELSE
        WRITELN('PLEASE ANSWER THE QUESTION WITH YES OR NO.');
  END; (* YNINFO *)

PROCEDURE RESURRECT1;
  BEGIN (* RESURRECT1 *)
    WITH WORDPTR↑ DO
      IF MEANING = ACT THEN
        IF ACTVAL = NO THEN BEGIN
          WRITELN('VERY WELL.');
          DONE := TRUE; END
        ELSE IF ACTVAL = YES THEN BEGIN
          QUESTION := NOQUEST;
          CASE NUMDIED OF
            1 : BEGIN
                  WRITELN('GREAT!  WHERE DID I PUT MY MAGIC     ');
                  WRITELN('DUST?  AH...  HERE IT IS.  I''LL     ');
                  WRITELN('SPRINKLE SOME DUST OVER YOU AND      ');
                  WRITELN('    .   .   .   > POOF <   .   .   . ');
                  WRITELN('THE ROOM DISAPPEARS IN A CLOUD OF    ');
                  WRITELN('GREEN SMOKE AND WHEN THE AIR CLEARS  ');
                  WRITELN('YOU FIND YOURSELF ...                ');
                  RESTART; END;
            2 : BEGIN
                  WRITELN('WHERE DID I PUT MY MAGIC DUST?  WOW, ');
                  WRITELN('THERE IS JUST A LITTLE BIT LEFT.     ');
                  WRITELN('I''LL SPRINKLE WHAT''S LEFT AND      ');
                  WRITELN('    .   .   .   > PUFF <   .   .   . ');
                  WRITELN('THE ROOM FADES AWAY IN A GREEN HAZE  ');
                  WRITELN('AND ...                              ');
                  RESTART; END;
            3 : BEGIN
                  WRITELN('I''LL LEAVE YOU ONE HINT BEFORE I LET');
                  WRITELN('YOU TRY IT YOURSELF.  YOU''LL NEED A ');
                  WRITELN('POWERFUL MAGIC WORD.  GOODBYE!       ');
                  QUESTION := LASTCHANCE; END;
          END; END
        ELSE IF ACTVAL = QUIT THEN
          DONE := TRUE
        ELSE
          WRITELN('PLEASE ANSWER THE QUESTION WITH YES OR NO.')
      ELSE
        WRITELN('PLEASE ANSWER THE QUESTION WITH YES OR NO.');
  END; (* RESURRECT1 *)

PROCEDURE RESURRECT2;
  BEGIN (* RESURRECT2 *)
    WITH WORDPTR↑ DO BEGIN
      DONE := TRUE;
      IF MEANING = DIRECT THEN
        IF DIRVAL = MAGIC THEN
          IF STRING[1] = 'Z' THEN BEGIN
```

```
                WRITELN('CONGRATULATIONS;  YOU DID IT.');
                DONE := FALSE;
                RESTART; END
              ELSE
                WRITELN('SEE, ONLY I HAVQ7UM<ICIENT MAGIC.')
            ELSE
              WRITELN('SEE, ONLY I HAVE SUJFICIENT MAGIC.')
          ELSE
            WRITELN('SEE, ONLY I HAVE SUFFICIENT MAGIC.'); END;
      END; (* RESURRECT2 *)

    PROCEDURE YNQUIT;
      BEGIN (* YNQUIT *)
        WITH WORDPTR↑ DO
          IF MEANING = ACT THEN
            IF ACTVAL = NO THEN BEGIN
              WRITELN('OK.');
              QUESTION := NOQUEST; END
            ELSE IF ACTVAL = YES THEN BEGIN
              WRITELN('VERY WELL.');
              QUESTION := NOQUEST;
              DONE := TRUE; END
            ELSE
              WRITELN('PLEASE ANSWER THE QUESTION WITH YES OR NO.')
          ELSE
            WRITELN('PLEASE ANSWER THE QUESTION WITH YES OR NO.');
        END; (* YNQUIT *)

    BEGIN (* ANALYZEQUESTION *)
      CASE QUESTION OF
        NOQUEST     : HALT;
        INFOQUEST   : YNINFO;
        DEADQUEST   : RESURRECT1;
        LASTCHANCE  : RESURRECT2;
        QUITQUEST   : YNQUIT;
      END;    END; (* ANALYZEQUESTION *)

  BEGIN (* QUERYHUMAN *)
 9: (* DON''T UNDERSTAND LOOP *)
    READLINE;
    COMMANDS := COMMANDS + 1;
    READTOKEN;
    IF WORDPTR = VOID THEN
      BLEWIT;
    IF QUESTION = NOQUEST THEN
      CASE WORDPTR↑.MEANING OF
        DIRECT   : DOMOVEMENT;
        ACT      : DOACTION;
        KNOWN    : BLEWIT;
        LOCATE   : BLEWIT;
        UNKNOWN  : BLEWIT;
      END
    ELSE
      ANALYZEQUESTION;
  END; (* QUERYHUMAN *)

ROCEDURE STAGEUNIVERSE;
   VAR I : INTEGER;

 PROCEDURE THROW ( MISS : BOOLEAN );
   BEGIN (* THROW *)
     WITH WHERE↑ DO
       IF (RANDOM(O) < 0.5) OR MISS THEN
         IF NOT (KNIFE IN CARRY) THEN BEGIN
```

```
          IF KNIFE IN PRESENT THEN BEGIN
            WRITE  ('THE ORC PICKS UP THE KNIFE ');
            WRITELN('AND THROWS IT AT YOU.'); END
          ELSE BEGIN
            PRESENT := PRESENT + [KNIFE];
            WRITELN('THE ORC THROWS A KNIFE AT YOU.'); END;
          WRITELN;
          IF (RANDOM(0) < 0.10) AND (NOT MISS) THEN BEGIN
            WRITELN('IT HITS YOU!');
            OOPS; END
          ELSE
            WRITELN('IT MISSES YOU!');
          IF RANDOM(0) < 0.25 THEN BEGIN
            PRESENT := PRESENT - [ORC];
            WRITELN;
            WRITELN('STRANGE, THE ORC JUST RAN AWAY.');
            NUMORC := NUMORC + 1;
            SAFORC := ORCSAFE; END; END;
    END; (* THROW *)

  BEGIN (* STAGEUNIVERSE *)
    WITH WHERE↑ DO BEGIN
      IF WAS <> NIL THEN
        IF WHERE <> WAS THEN
          IF (CLASS = LAND) AND (SIDE = INSIDE) AND LIGHT THEN
            IF ORC IN WAS↑.PRESENT THEN
              IF NOT (ORC IN PRESENT) THEN BEGIN
                WAS↑.PRESENT := WAS↑.PRESENT - [ORC];
                PRESENT := PRESENT + [ORC]; END;
      IF KNIFE IN CARRY THEN
        IF KNIFE IN PRESENT THEN
          PRESENT := PRESENT - [KNIFE];
      IF QUESTION = NOQUEST THEN BEGIN
        IF WHERE <> WAS THEN
          TELLLOCATION(BRIEFLY);
        IF (WHERE = WAS) AND (WARN = 14) THEN BEGIN
          WRITE  ('YOU''VE STOOD TOO LONG ON THE ');
          WRITELN('LAVA AND COOKED YOUR GOOSE.');
          OOPS; END;
        IF KNIFE IN CARRY THEN
          IF (WET = WATERWET) OR (CLASS <> LAND) OR
             (POOL = WATERPOOL) THEN BEGIN
            CARRY := CARRY - [KNIFE];
            WRITELN('THE MOIST AIR CAUSES THE KNIFE TO DISCOLOR AND');
            WRITELN('CRUMBLE INTO DUST.'); END;
        IF ([DRAGON,SNAKE]*PRESENT) <> [] THEN BEGIN
          IF RIGHTECH THEN
            RIGHTECH := FALSE
          ELSE
            STRANGLE := STRANGLE + 1;
          I := STRANGLE DIV 7;
          IF KNIFE IN CARRY THEN BEGIN
            STRANGLE := 21;
            I := 3;
            WRITELN;
            IF DRAGON IN PRESENT THEN
              WRITELN('THE DRAGON SEES YOUR KNIFE')
            ELSE
              WRITELN('THE SNAKE SEES YOUR KNIFE');
            WRITELN('AND BECOMES EXTREMELY ANGRY.'); END;
          IF STRANGLE = 7*I THEN BEGIN
            WRITELN;
            CASE I OF
              1: IF DRAGON IN PRESENT THEN BEGIN
              WRITE  ('THE DRAGON SINGES YOUR ');
              WRITELN('HAIR WITH HIS BREATH.'); END
            ELSE
              WRITELN('THE SNAKE COILS MENANCINGLY AROUND YOU.');
         2: IF DRAGON IN PRESENT THEN BEGIN
              WRITE  ('THE DRAGON SCORCHES YOUR ');
              WRITELN('CLOTHES WITH HIS BREATH.'); END
            ELSE
              WRITELN('THE SNAKE COILS TIGHTLY AROUND YOU.');
         3: BEGIN
              IF DRAGON IN PRESENT THEN
                WRITELN('THE DRAGON ROASTS YOU TO A CINDER.')
              ELSE BEGIN
                WRITE  ('THE SNAKE SQUEEZES AND ');
                WRITELN('CRUSHES YOUR BONES.'); END;
              OOPS; END;
      END; END; END;
  WAS := WHERE;
  WITH NOUNS[LAMP] DO
    IF BURNING THEN BEGIN
      TIMELEFT := TIMELEFT - 1;
      IF TIMELEFT = O THEN BEGIN
        BURNING := FALSE;
        IF LAMP IN (CARRY + PRESENT) THEN BEGIN
          WRITELN('THE LANTERN JUST RAN OUT OF FUEL.');
          WRITELN('YOU''LL NEED OIL TO REFILL IT AND A MATCH');
          WRITELN('TO LIGHT IT.'); END; END
      ELSE IF TIMELEFT = 15 THEN
        IF LAMP IN (CARRY + PRESENT) THEN BEGIN
          WRITELN('THE LANTERN IS RUNNING LOW ON FUEL.  YOU');
          WRITELN('MAY BE ABLE TO FILL IT WITH SOME OIL.');
          END; END;
  IF ORC IN PRESENT THEN
    THROW(FALSE)
  ELSE
    IF (CLASS = LAND) AND (SIDE = INSIDE) THEN BEGIN
      IF PSTATE = PACTIVE THEN
        IF RANDOM(O) < O.0633 THEN
          IF RANDOM(O) < O.3333 THEN BEGIN
            WRITE  ('A PIRATE APPEARS OUT OF THE ');
            WRITELN('DARKNESS FOR JUST A MOMENT.');
            IF (TREAS*CARRY) <> [] THEN BEGIN
              WRITE  ('WHILE HE''S HERE HE SAYS:  ''I''LL TAKE ');
              WRITELN('YOUR TREASURES AND HIDE THEM AWAY.');
              WITH NOUNS[CHEST] DO BEGIN
                CHESTCONTENTS := CHESTCONTENTS + TREAS*CARRY;
                CHESTSTATE := LOCKEDCHEST; END;
              CARRY := CARRY - TREAS*CARRY; END; END
          ELSE BEGIN
            WRITE  ('A PIRATE RUNS BY, POKES YOU IN THE ');
            WRITELN('RIBS, LAUGHS, AND DISAPPEARS.'); END;
      IF SAFORC > O THEN
        SAFORC := SAFORC - 1
      ELSE
        IF NUMORC > O THEN
          IF RANDOM(O) < O.1 THEN BEGIN
            NUMORC := NUMORC - 1;
            IF (NUMORC = (ORCNUMBER DIV 2))
               AND (PSTATE = PWAIT) THEN
              PSTATE := PACTIVE;
            SAFORC := ORCSAFE;
            WRITELN('AN UGLY AND MEAN ORC HAS FOUND YOU.');
            IF NUMORC = ORCNUMBER-1 THEN BEGIN
              PRESENT := PRESENT + [AXE];
```

```
                  WRITELN('THE ORC THROWS AN AXE AT YOU.');
                  WRITELN('IT MISSES YOU!');
                  WRITELN('STRANGE, THE ORC JUST RAN AWAY.'); END
                ELSE BEGIN
                  PRESENT := PRESENT + [ORC];
                  THROW(TRUE); END; END; END; END; END;
 END; (* STAGEUNIVERSE *)

EGIN (* ADVENTURE *)
 INITIALIZE;
 REPEAT
```

```
O: STAGEUNIVERSE;ON IMMEDIATELY *)
   IF NOT DONE THEN
     QUERYHUMAN;
 UNTIL DONE;
 SCOREGAME;
ND. (* ADVENTURE *)
```

```
         IDENT   GETFIL
         ENTRY   GETFIL

ADVENT4  FILEB   GETFIL,1,(PFN=ADVDATA),(USN=TPA),(PWD=TOM)

GETFIL   PS
         SYSTEM TLX,P,70B    DISABLE TERMINAL CONTROL
```

```
ATTACH ADVENT4,,,,R
EQ     GETFIL
END
```