```
    {1992 ACM Mid-Central Regional Programming Contest
 Sample Solution to Problem #1 - Puttin' on the Hex}

{Turbo Pascal 6.0}
{$A+,B-,D+,E-,F-,I+,L+,N-,O-,R+,S+,V+}
{$M 16384,0,655360}

PROGRAM HexMaze;

CONST
   p : INTEGER = 0;
   m : ARRAY[ 1..9, 1..9 ] OF INTEGER =
      (( 4, 1, 2, 2, 4, 0, 0, 0, 0 ),
       ( 4, 4, 1, 1, 3, 2, 0, 0, 0 ),
       ( 1, 5, 3, 3, 4, 2, 2, 0, 0 ),
       ( 5, 2, 3, 2, 3, 1, 2, 1, 0 ),
       ( 4, 3, 2, 1, 3, 3, 2, 3, 2 ),
       ( 2, 5, 2, 4, 2, 3, 1, 1, 0 ),
       ( 2, 2, 2, 2, 4, 2, 6, 0, 0 ),
       ( 4, 3, 1, 3, 1, 3, 0, 0, 0 ),
       ( 2, 2, 2, 4, 2, 0, 0, 0, 0 ));

VAR
   s : ARRAY[ 1..1000 ] OF RECORD d, r, c : INTEGER END;
   f : TEXT;

(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)

PROCEDURE Push( d, r, c : INTEGER );

   BEGIN
   INC( p );
   s[ p ].d := d;
   s[ p ].r := r;
   s[ p ].c := c
   END; { Push }

(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)

PROCEDURE Pop;

   BEGIN
   DEC( p )
   END; { Pop }

(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)

PROCEDURE ShowSolution;

   VAR
      i : INTEGER;

   BEGIN
   WriteLn( f, ' MOVE      MOVE     POSITION' );
   WriteLn( f, 'NUMBER     TYPE     row  hex' );
   WriteLn( f, '------     ----     ---  ---' );
   WriteLn( f, '   0        GO       5    5 ' );
   Write( f, '   1       3' );
   FOR i := 1 TO p DO
      BEGIN
```

```
      IF i <> 1 THEN Write( f, i:4, m[ s[ i - 1 ].r, s[ i - 1 ].c ]:9 );
      CASE s[ i ].d OF
         0 : Write( f, 'W ' );
         1 : Write( f, 'NW' );
         2 : Write( f, 'NE' );
         3 : Write( f, 'E ' );
         4 : Write( f, 'SE' );
         5 : Write( f, 'SW' )
         END;
       WriteLn( f, s[ i ].r:7, s[ i ].c:5 )
       END
   END; { ShowSolution }

(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)

FUNCTION Lost( d, r, c : INTEGER ) : BOOLEAN;

   VAR
      l : BOOLEAN;
      i : INTEGER;

   BEGIN
   l := FALSE;
   i := 1;
   WHILE (NOT l) AND (i < p) DO
      BEGIN
      IF (s[ i ].d = d) AND (s[ i ].r = r) AND (s[ i ].c = c) THEN l := TRUE;
      INC( i )
      END;
   Lost := l
   END; { Lost }

(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)

PROCEDURE MoveOneHex( d, r, c : INTEGER; VAR i, j : INTEGER );

   BEGIN
   CASE d OF
      0 : BEGIN (* West *)
          i := r;
          j := c - 1
          END;
      1 : BEGIN (* Northwest *)
          i := r - 1;
          IF r > 5 THEN j := c ELSE j := c - 1
          END;
      2 : BEGIN (* Northeast *)
          i := r - 1;
          IF r < 6 THEN j := c ELSE j := c + 1
          END;
      3 : BEGIN (* East *)
          i := r;
          j := c + 1
          END;
      4 : BEGIN (* Southeast *)
          i := r + 1;
          IF r > 4 THEN j := c ELSE j := c + 1
          END;
      5 : BEGIN (* Southwest *)
          i := r + 1;
```

```
        IF r < 5 THEN j := c ELSE j := c - 1
        END
      END
   END; { MoveOneHex }
(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)
PROCEDURE FindHex( d, r, c : INTEGER; VAR i, j : INTEGER );

   VAR
      k : INTEGER;

   BEGIN
   i := r;
   j := c;
   FOR k := 1 TO m[ r, c ] DO MoveOneHex( d, i, j, i, j )
   END; { FindHex }
(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)
FUNCTION HexValue( r, c : INTEGER ) : INTEGER;

   BEGIN
   IF (r < 1) OR (r > 9) OR (c < 1) OR (c > 9) THEN
      HexValue := 0
   ELSE
      HexValue := m[ r, c ]
   END; { HexValue }
(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)
FUNCTION Solve( d, r, c : INTEGER ) : BOOLEAN;

   VAR
      i, j, k : INTEGER;
      s       : BOOLEAN;

   BEGIN
   Push( d, r, c );
   IF (r = 5) AND (c = 5) THEN
      BEGIN
      ShowSolution;
      Solve := TRUE
      END
   ELSE IF Lost( d, r, c ) THEN
      Solve := FALSE
   ELSE
      BEGIN
      (* Try turning left. *)
      IF d = 0 THEN k := 5 ELSE k := d - 1;
      FindHex( k, r, c, i, j );
      IF HexValue( i, j ) <> 0 THEN s := Solve( k, i, j ) ELSE s := FALSE;
      (* Try continuing forward. *)
      IF NOT s THEN
         BEGIN
         FindHex( d, r, c, i, j );
         IF HexValue( i, j ) <> 0 THEN s := Solve( d, i, j )
         END;
      (* Try turning right. *)
      IF NOT s THEN
```

```
         BEGIN
         IF d = 5 THEN k := 0 ELSE k := d + 1;
         FindHex( k, r, c, i, j );
         IF HexValue( i, j ) <> 0 THEN s := Solve( k, i, j )
         END;
      Solve := s
      END;
   Pop
   END; { Solve }

(* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)

BEGIN
Assign( f, 'HEX.OUT' );
Rewrite( f );
IF NOT Solve( 0, 5, 2 ) THEN
   IF NOT Solve( 1, 2, 2 ) THEN
      IF NOT Solve( 2, 2, 5 ) THEN
         IF NOT Solve( 3, 5, 8 ) THEN
            IF NOT Solve( 4, 8, 5 ) THEN
               IF NOT Solve( 5, 8, 2 ) THEN
                  WriteLn( f, 'No solution possible.' );
Close( f )
END. { HexMaze }
```