```
    (1992 ACM Mid-Central Regional Programming Contest
 Sample Solution to Problem #7 - Amazing Similarities)

(* AMAZING.PAS - Turbo Pascal Version 6.0 *)
{$A+,B-,D+,E-,F-,I+,L+,N-,O-,R+,S+,V+}
{$M 65520,0,655360}

(* This problem reduces to determining if two trees are structurally
   identical (i.e. isomorphic).  There is a well-known linear-time algorithm
   for accomplishing this test, and several quadratic-time algorithms which
   are relatively easy to discover and implement.  The maximum dimensions of
   the input mazes (10 by 10) have been selected to allow a quadratic
   algorithm to run successfully within the problem time limits.

   The quadratic solution implemented here involves computing the path
   lengths from every leaf to every other leaf.  If two trees are
   structurally equivalent, then this set of lengths must be identical. *)

PROGRAM Amazing;

CONST
   MAXDIM = 10;

TYPE
   MAZE = ARRAY[ 1..MAXDIM, 1..MAXDIM ] OF CHAR;

VAR
   input, output  : TEXT;

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

FUNCTION Homogeneous( VAR m1, m2 : MAZE; r1, r2, c1, c2 : INTEGER ) : BOOLEAN;

   TYPE
      LIST    = ^LISTREC;
      LISTREC = RECORD n : INTEGER; f : LIST END;

   FUNCTION MazeVal( VAR m : MAZE; r, c, i, j : INTEGER ) : CHAR;

      BEGIN
      IF (i < 1) OR (i > r) OR (j < 1) OR (j > c) THEN
         MazeVal := 'X'
      ELSE
         MazeVal := m[ i, j ]
      END; { MazeVal }

   FUNCTION Neighbors( VAR m : MAZE; r, c, i, j : INTEGER ) : INTEGER;

      VAR
         n : INTEGER;

      BEGIN
      n := 0;
      IF MazeVal( m, r, c, i - 1, j ) = '.' THEN INC( n );
      IF MazeVal( m, r, c, i, j + 1 ) = '.' THEN INC( n );
      IF MazeVal( m, r, c, i + 1, j ) = '.' THEN INC( n );
      IF MazeVal( m, r, c, i, j - 1 ) = '.' THEN INC( n );
      Neighbors := n
      END; { Neighbors }
```

```
PROCEDURE Insert( VAR l : LIST; n : INTEGER );

   VAR
      r, p, c : LIST;

   BEGIN
   NEW( r );
   r^.n := n;
   IF l = NIL THEN
      BEGIN
      r^.f := NIL;
      l := r
      END
   ELSE
      BEGIN
      p := NIL;
      c := l;
      WHILE (c <> NIL) AND (r^.n > c^.n ) DO
         BEGIN
         p := c;
         c := c^.f
         END;
      IF p <> NIL THEN
         BEGIN
         r^.f := c;
         p^.f := r
         END
      ELSE
         BEGIN
         r^.f := l;
         l := r
         END;
      END
   END; { Insert }

FUNCTION EqList( l1, l2 : LIST ) : BOOLEAN;

   VAR
      eq : BOOLEAN;

   BEGIN
   eq := TRUE;
   WHILE eq AND (l1 <> NIL) AND (l2 <> NIL) DO
      BEGIN
      eq := l1^.n = l2^.n;
      l1 := l1^.f;
      l2 := l2^.f
      END;
   EqList := eq AND (l1 = NIL) AND (l2 = NIL)
   END; { EqList }

PROCEDURE FreeList( l : LIST );

   VAR
      d : LIST;

   BEGIN
   d := l;
   WHILE d <> NIL DO
      BEGIN
```

```
      l := l^.f;
      DISPOSE( d );
      d := l
      END
   END; { FreeList }

PROCEDURE DFS( VAR m : MAZE; r, c, i, j, n : INTEGER; VAR l : LIST );

   BEGIN
   m[ i, j ] := '@';
   IF Neighbors( m, r, c, i, j ) = 0 THEN
      Insert( l, n )
   ELSE
      BEGIN
      IF MazeVal( m, r, c, i - 1, j ) = '.' THEN
         DFS( m, r, c, i - 1, j, n + 1, l );
      IF MazeVal( m, r, c, i, j + 1 ) = '.' THEN
         DFS( m, r, c, i, j + 1, n + 1, l );
      IF MazeVal( m, r, c, i + 1, j ) = '.' THEN
         DFS( m, r, c, i + 1, j, n + 1, l );
      IF MazeVal( m, r, c, i, j - 1 ) = '.' THEN
         DFS( m, r, c, i, j - 1, n + 1, l )
      END;
   m[ i, j ] := '.'
   END; { DFS }

VAR
   i, j   : INTEGER;
   l1, l2 : LIST;

BEGIN
l1 := NIL;
l2 := NIL;
FOR i := 1 TO MAXDIM DO
   FOR j := 1 TO MAXDIM DO
      BEGIN
      IF (MazeVal( m1, r1, c1, i, j ) = '.') AND (Neighbors( m1, r1, c1, i, j
         DFS( m1, r1, c1, i, j, 0, l1 );
      IF (MazeVal( m2, r2, c2, i, j ) = '.') AND (Neighbors( m2, r2, c2, i, j
         DFS( m2, r2, c2, i, j, 0, l2 );
      END;
Homogeneous := EqList( l1, l2 );
FreeList( l1 );
FreeList( l2 );
END; { Homogeneous }

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

FUNCTION ReadMaze( VAR m : MAZE; VAR r, c : INTEGER; VAR id : STRING ) : BOOLEAN

VAR
   ch   : CHAR;
   i, j : INTEGER;

BEGIN
ReadLn( input, r, c, ch, id );
IF r <> 0 THEN
 BEGIN
   FOR i := 1 TO r DO
      BEGIN
```

```
         FOR j := 1 TO c DO Read( input, m[ i, j ] );
         ReadLn( input )
         END;
      ReadMaze := TRUE
    END
   ELSE
      ReadMaze := FALSE
   END; { ReadMaze }

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

VAR
   m1, m2             : MAZE;
   r1, c1, r2, c2 : INTEGER;
   id1, id2          : STRING;

BEGIN
Assign( input, 'amazing.in' );
Reset( input );
Assign( output, 'amazing.out' );
Rewrite( output );
WHILE ReadMaze( m1, r1, c1, id1 ) DO
   BEGIN
   if ReadMaze( m2, r2, c2, id2 ) then ;
   Write( output, '"', id1, '" and "', id2, '" ARE' );
   IF NOT Homogeneous( m1, m2, r1, r2, c1, c2 ) THEN Write( output, ' NOT' );
   WriteLn( output, ' homogeneous' )
   END;
Close( input );
Close( output )
END. { Amazing }
```