CS19002 PDS Lab, Test 2, Apr 03, 2009

(For students with odd PC numbers)

You are given a 20×20 matrix, some cells of which are marked *dangerous*. Your task is to find a path from the $(0, 0)$-th cell to the $(19, 19)$-th cell by avoiding all the dangerous cells. Assume that these source and destination cells are not dangerous. Your path need not be the shortest safe path. Reporting any path will do. However, there must not be any repetition of cells on the path.

Write a ***non-recursive*** function to solve this problem.

Follow a strategy as described now. Suppose that you are currently sitting at the (i, j)-th cell of the matrix. Also assume that the current cell is not dangerous. From this cell, you have four possible movements: $(i, j+1)$ (right), $(i+1, j)$ (down), $(i, j-1)$ (left) and $(i-1, j)$ (up). (Diagonal movements are not allowed.) Consider all these possibilities one by one. If your movement leads to a dangerous or visited cell (or outside the boundary of the matrix), discard the movement. Otherwise, continue the exploration. If all directions are already explored from the current cell and you cannot reach the destination cell from the current cell, reject the current cell and move back to the cell from which you stepped into the (i, j)-th cell. Note that you would possibly like to maintain a two-dimensional array, the (i, j)-th entry of which remembers the directions that are already explored from the (i, j)-th cell.

You are provided a skeleton of the program overleaf.

Part 1: Complete the `printmat()` function in order to obtain an output as shown below. Dangerous cells are marked by `X` and non-dangerous cells by `.` (dot). **(25%)**

Part 2: Write the path-finding function mentioned above, and call that function from `main()` with appropriate argument values. **(75%)**

Sample outputs of your program for 10×10 matrices are shown below.

```
. . . . . X . . . .
. . X X . . . X X .
. . . . X X . X . .
X X . . . . . . X .
. X . . X . . . . .
. . . . X . . . . .
X . . . . . X . . X
. . X . . . . . X X
. . . X X . . . X .
X X . . . . . . . .

Path found:
( 0, 0) ( 0, 1) ( 0, 2) ( 0, 3) ( 0, 4) ( 1, 4) ( 1, 5) ( 1, 6) ( 2, 6) ( 3, 6)
( 3, 7) ( 4, 7) ( 4, 8) ( 4, 9) ( 5, 9) ( 5, 8) ( 6, 8) ( 6, 7) ( 7, 7) ( 8, 7)
( 9, 7) ( 9, 8) ( 9, 9)

. . . . . . . . . X
X X . X . . . . . .
. . . . X X X . . .
. . . X X . . . X .
. X . . . . X . X .
. X . X . . X . . .
X . X . . X . . . .
. . . . . . . . X .
. . . . X X . X . X
. . . . . X . X . .

No path found...
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define DIM 20
#define DANGER 1
#define NODANGER 2
#define VISITED 3

void initmat ( int A[DIM][DIM] )
{
   int i, j;

   for (i=0; i<DIM; ++i) for (j=0; j<DIM; ++j)
      A[i][j] = (rand() % 4) ?  NODANGER : DANGER;
   A[0][0] = A[DIM-1][DIM-1] = NODANGER;
}

void printmat ( int A[DIM][DIM] )
{
   /*** COMPLETE THIS FUNCTION SO AS TO OBTAIN AN OUTPUT AS SHOWN EARLIER ***/
}

/*** WRITE THE PATH-FINDING FUNCTION HERE ***/

int main ()
{
   int A[DIM][DIM];

   srand((unsigned int)time(NULL));
   initmat(A);
   printmat(A);

   /*** CALL THE PATH-FINDING FUNCTION HERE ***/

   exit(0);
}
```

CS19002 PDS Lab, Test 2, Apr 03, 2009

(For students with <u>even</u> PC numbers)

You are given a 20×20 matrix, some cells of which are marked *dangerous*. Your task is to find a path from the $(0, 0)$-th cell to the $(19, 19)$-th cell by avoiding all the dangerous cells. Assume that these source and destination cells are not dangerous. Your path need not be the shortest safe path. Reporting any path will do. However, there must not be any repetition of cells on the path.

Write a ***<u>recursive</u>*** function to solve this problem. Since your recursive function would print a path backward, *after* the entire path is discovered, you would like to find a path from $(19, 19)$ to $(0, 0)$.

Follow a strategy as described now. Suppose that you are currently sitting at the (i, j)-th cell of the matrix. Also assume that the current cell is not dangerous. From this cell, you have four possible movements: $(i, j+1)$ (right), $(i+1, j)$ (down), $(i, j-1)$ (left) and $(i-1, j)$ (up). (<u>Diagonal movements are not allowed.</u>) Recursively consider only those possibilities that do not lead you to dangerous or visited cells or outside the boundary of the matrix. If any of these recursive calls gives you a path from (i, j) to $(19, 19)$, you are through. If all the recursive calls fail, then there is no way to reach $(19, 19)$ from (i, j).

You are provided a skeleton of the program overleaf.

Part 1: Complete the `printmat()` function in order to obtain an output as shown below. Dangerous cells are marked by `X` and non-dangerous cells by `.` (dot). **(25%)**

Part 2: Write the path-finding function mentioned above, and call that function from `main()` with appropriate argument values. **(75%)**

Sample outputs of your program for 10×10 matrices are shown below.

```
. . . . . X . . . .
. . X X . . . X X .
. . . . X X . X . .
X X . . . . . . X .
. X . . X . . . . .
. . . . X . . . . .
X . . . . . X . . X
. . X . . . . . X X
. . . X X . . . X .
X X . . . . . . . .

Path found:
( 0, 0) ( 0, 1) ( 0, 2) ( 0, 3) ( 0, 4) ( 1, 4) ( 1, 5) ( 1, 6) ( 2, 6) ( 3, 6)
( 3, 7) ( 4, 7) ( 4, 8) ( 4, 9) ( 5, 9) ( 5, 8) ( 6, 8) ( 6, 7) ( 7, 7) ( 8, 7)
( 9, 7) ( 9, 8) ( 9, 9)

. . . . . . . . . X
X X . X . . . . . .
. . . . X X X . . .
. . . X X . . . X .
. X . . . . X . X .
. X . X . . X . . .
X . X . . X . . . .
. . . . . . . . X .
. . . . X X . X . X
. . . . . X . X . .

No path found...
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define DIM 20
#define DANGER 1
#define NODANGER 2
#define VISITED 3

void initmat ( int A[DIM][DIM] )
{
   int i, j;

   for (i=0; i<DIM; ++i) for (j=0; j<DIM; ++j)
      A[i][j] = (rand() % 4) ?  NODANGER : DANGER;
   A[0][0] = A[DIM-1][DIM-1] = NODANGER;
}

void printmat ( int A[DIM][DIM] )
{
   /*** COMPLETE THIS FUNCTION SO AS TO OBTAIN AN OUTPUT AS SHOWN EARLIER ***/
}

/*** WRITE THE PATH-FINDING FUNCTION HERE ***/

int main ()
{
   int A[DIM][DIM];

   srand((unsigned int)time(NULL));
   initmat(A);
   printmat(A);

   /*** CALL THE PATH-FINDING FUNCTION HERE ***/

   exit(0);
}
```