COMPUTER SCIENCE 241
Spring 2023

Programming Assignment 6

Due by end-of-day on Thursday, April 13

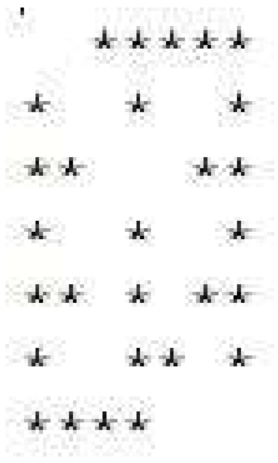
Recursion is a very useful tool to solve many kinds of problems that exhibit a "brute-force, try everything" solution strategy. For this assignment, you must write a recursive program that will find a route through a maze. The maze will be represented by a 2-dimensional array of characters, as illustrated below by the 7-by-7 maze below:

```
  *****
*  *  *
**   **
*  * **
** *  *
*  ** *
****
```

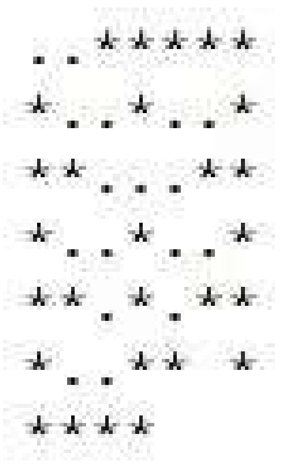
Each cell of the maze is either a '*' (which represents a wall) or a blank space ' ' (which represents a passageway). You should assume that the person starts out in the upper-left corner of the maze, and wants to find a route to the lower-right corner of the maze. Your program should search for a route, and either report that no route exists, or if there is a route, your program should place an 'X' at each step along the route. For example, the input maze on the left below has a solution, and the output of your program should be similar to the maze shown on the right below. You do **not** need to find a shortest route, only a possible route.

```
  *****          XX*****
*  *  *          *XX*  *
**   **          **XXX**
*  * **          *..*X**
** *  *          **.*XX*
*  ** *          *..**X*
****             **** XX
```

The input maze on the left below has no solution, and the output of your program should match the maze shown on the right below, which indicates that the program visited every possible reachable passageway before it “gave up”



```
  *****
*  *  *
**   **
*  *  *
** * **
*  ** *
****
```



```
..*****
*..*..*
**...**
*..*..*
**.*.**
*..** *
****
```

This route-finding problem is naturally recursive. At any point in time, your program can continue to search for the route by trying each of the 4 possible directions from the current location. Of course, your code should check that you do not go out-of-bounds of the grid, and that you do not travel through a wall.

Notice that the maze may contain cycles. Your program should not get caught in an endless loop, going around and around a circle in the maze. To prevent this, your program should place a dot character ‘.’ at each cell in the maze which your program has visited, but which did not lead to a solution. Your program **must** display these dots. Your program should no “erase” these dots before displaying the final solution.

You should **modify** the code at: ~joanmlucas/cs241/programs/prog06/handouts/maze.c

You must add and call a recursive function that will locate a route through the maze (if it exists). This program reads in the maze from standard input. Since we do not wish to type in the maze each time, you should run your program as:

```
$ ./a.out  <   testMaze1
```

Several test data files, that contain the input maze, are provided in the **handouts** directory.

Your program should return an **exit status** of 0 if the route was successfully found. Otherwise, your program should return an exit status of 1.

You should be sure to include **YOUR NAME** in a comment at the top of your source code file. Your source code must use proper **style**, i.e., variables should be well named (name is not too short, not too long, and is meaningful), and bodies of loops, if’s, etc.. should be properly indented.

You should submit your program for grading by copying it into my account using the following command (assuming your name is J. Smith).

```
$ cp   smith_j_prog06.c    ~joanmlucas/cs241/programs/prog06/submission
```

The timestamp of this copied file will indicate when you submitted this assignment. **After** you have copied the file, you must **be sure to set the permissions** on this copied file as follows:

```
$ chmod  644  ~joanmlucas/cs241/programs/prog06/submission/smith_j_prog06.c
```

You can verify that your file was successfully copied using the ls (list) command:

```
$ ls -l  ~joanmlucas/cs241/programs/prog06/submission/smith_j_prog06.c
```