CSCI-15 Recursion programming assignment #8, (80 points), due 12/6/18

Write a program to read a maze from a text file into your array, print the unsolved maze, solve the maze, and then print the solution. You may assume the maze will fit in a 24-row by 81-column character array (for 80 character C-strings). The maze will be in a file with the number of rows and columns on the first line, followed by the lines defining the maze, with `'*'` representing a wall and `' '` (space) representing a corridor. For example, here is a small (8-by-12) maze (on my web site as maze8-12.txt):

```
8 12
  **********
* *        *
*   ****** *
*****      *
*     ******
*** ***    *
*       ** *
**********
```

You may assume that the maze will always have a solution. The program is easily generalized to show if there is no solution, but this is not required for this assignment. For an n-by-m maze held in a char array a[NRows][NCols], NRows >= n and NCols >= m, the starting point (at the top left) is a[0][0], and the ending point (at the bottom right) is at a[n-1][m-1]. When you print the solved maze, show the path through the maze by marking all locations in the solution path with `'#'`. For example, here is the same maze printed showing the solution:

```
##**********
*#*#########*
*###*******#*
*****######*
*  ###******
***#***####*
*  ######*#*
**********##
```

Notice, moves down "blind alleys" are left off the solution. I will discuss the algorithm for this in class, and you must use the recursive backtracking algorithm I give you for this program. This is not a large program – no more than about two pages of code.

Test with the three mazes I supply on my Web site and a few of your own making. Include the input and output maze files with your tests. All I/O (except for file access error messages) will be to files directly from within your program. You may prompt for the file names or take them from the command line, but you may not hard-code them in your program.