60-141 – Introduction to Programming II Winter, 2017
Assignment 2
(Deadline Sunday, Feb. 12, 2017 before 23:59)

Maze Traversal

The following grid is a double-subscripted array representation of a maze.

```
# # # # # # # # # # # #
# . . . # . . . . . . #
. . # . # . # # # # . #
# # # . # . . . . # . #
# . . . . # # # . # . .
# # # # . # . # . # . #
# . . # . # . # . # . #
# # . # . # . # . # . #
# . . . . . . . . # . #
# # # # # # . # # # . #
# . . . . . . # . . . #
# # # # # # # # # # # #
```

The # symbols represent the walls of the maze, and the periods (.) represent squares in the possible paths through the maze. There's a simple algorithm for walking through a maze that guarantees finding the exit (assuming there's an exit). If there's not an exit, you'll arrive at the starting location again. Place your right hand on the wall to your right and begin walking forward. Never remove your hand from the wall. If the maze turns to the right, you follow the wall to the right. As long as you do not remove your hand from the wall, eventually you'll arrive at the exit of the maze. There may be a shorter path than the one you have taken, but you're guaranteed to get out of the maze.

For this assignment, the initial maze will use 1's instead of #'s and 0's instead of .'s, leading to the representation of the maze above in the form:

```
111111111111
100010000001
001010111101
111010000101
100001110100
111101010101
100101010101
110101010101
100000000101
111111011101
100000010001
111111111111
```

Your task is to:

Write a complete, well documented C program to walk through the maze, starting from the maze entrance location, and attempts to locate the exit from the maze.

Requirements and Hints:

1. Given the input maze, your program logic will:
   a. Find the maze entrance (start point, see Figure below) from the left side of the input maze (first column)
   b. Then, start to traverse the maze to find the next valid move
   c. If the movement is valid and it is not the maze exit (i.e. a coordinate on the maze edge, including the starting position)
      i. place character X in that location in the path
      ii. print the current state of the maze
      iii. Go back to step (b)
   d. Steps b. and c. will continue until finding the maze exit coordinate.
2. This solution assumes that there is only one entrance on the left edge of the maze and one exit on any other edges of the given maze (including the left edge). So, there are only two zeroes (one for entrance and one for exit) on the edges.
3. Your program should implement at least the following functions:
   a) void findStart(), that will find the start point from the first column of the input maze (maze[][1]).
   b) void mazeTraversal( ), is a **recursive** function that gives maze, current x and y coordinate and direction as an input. There are four valid directions of up, down, left and right. Also, the first (X,Y) coordinate to call mazeTraversal function is the found start point (maze entrance).
   c) void printMaze( ), that will print the current state of the maze after each movement.
   d) int validMove( ), that will determine the validity of the next movement (input coordinates).
   e) int coordsAreEdge( ), that check the input coordinate are edge or not.

Declaration of maze data, or Input:

A 12-by-12 character array representing the maze that has 1 symbols showing the walls of the maze and zeroes (0) showing squares in the possible paths through the maze. You can get the maze from a file or easily declare the maze data in main() of your program, manually.

Output

Placing the character X in each square in the path and displaying the maze after each move so the user can watch as the maze is solved. (Note that a move can revisit a square with an X in it.)

Sample declaration of maze data, or input:

```
char maze[12][12] ={{'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1'},
                    {'1', '0' ,'0' ,'0' ,'1', '0', '0' ,'0' ,'0' ,'0' ,'0' ,'1'},
                    …
                   }
```

You can complete the above two-dimensional maze array definition from the following Figure 1.

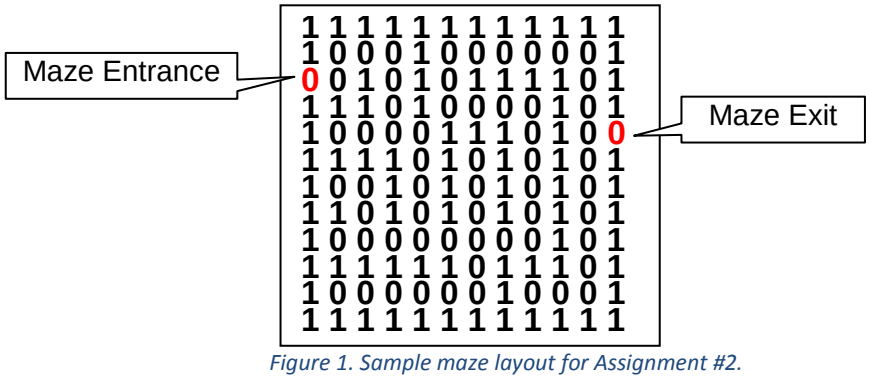

Figure 1. Sample maze layout for Assignment #2.

Sample output:

If you run your program in interactive mode, one move at a time, then the output sequence shown below is reasonable for the example maze shown in Figure 1:

```
...
1 1 1 1 1 1 1 1 1 1 1 1 1
1 X 0 0 1 0 0 0 0 0 0 1
X X 1 0 1 0 1 1 1 1 0 1
1 1 1 0 1 0 0 0 0 1 0 1
1 0 0 0 0 1 1 1 0 1 0 0
1 1 1 1 0 1 0 1 0 1 0 1
1 0 0 1 0 1 0 1 0 1 0 1
1 1 0 1 0 1 0 1 0 1 0 1
1 0 0 0 0 0 0 0 0 1 0 1
1 1 1 1 1 1 0 1 1 1 0 1
1 0 0 0 0 0 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1

Hit return to see next move

1 1 1 1 1 1 1 1 1 1 1 1 1
1 X X 0 1 0 0 0 0 0 0 1
X X 1 0 1 0 1 1 1 1 0 1
1 1 1 0 1 0 0 0 0 1 0 1
1 0 0 0 0 1 1 1 0 1 0 0
1 1 1 1 0 1 0 1 0 1 0 1
1 0 0 1 0 1 0 1 0 1 0 1
1 1 0 1 0 1 0 1 0 1 0 1
1 0 0 0 0 0 0 0 0 1 0 1
1 1 1 1 1 1 0 1 1 1 0 1
1 0 0 0 0 0 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1

Hit return to see next move
...
```

However, for generating the output from the program for submitting for marks, simply output the **final pathway** or failure to solve the maze (e.g. Entrance, but no Exit). This is shown below in Figure 2 for a successful maze traversal, using the strategy of touching one's right hand to the wall to the right; note that this solution is not unique – other correct solutions can be determined using various strategies:

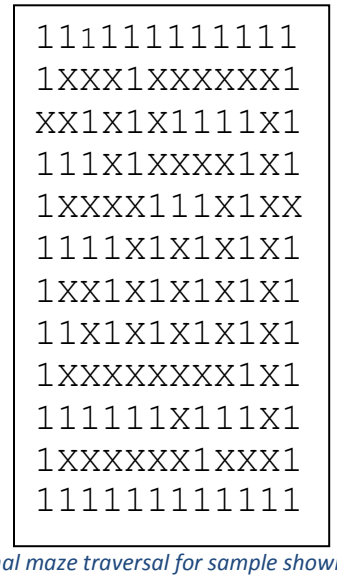

Figure 2. Final maze traversal for sample shown in Figure 1.

Requirements:

- Write and document a complete C program that is capable of satisfying the requirements of this assignment problem.
- UNDOCUMENTED OR IMPROPERLY DOCUMENTED code will automatically lose 50% marks.
- PLAGIARIZED work will not be graded and receive a mark of ZERO and reported according to the Senate bylaws.
- The question can use of I/O redirection. Please review the textbook for an example on using I/O redirection from flat files.
- TO SUBMIT: No later than the submission deadline, your assignment should be uploaded in Blackboard. Late submissions are not accepted and will receive a mark of ZERO.
- Upload your work as an attachment, include both the source file (assign2.c) and the script file (assign2.txt) - see below how to create the script file.

To create a script file (one that logs your compilation steps and your output in a text file):
1. script assign2.txt
2. cat assign2.c
3. cat input.txt
4. cc assign2.c
5. a.out < input.txt
6. ls -l
7. exit (DO NOT FORGET THIS STEP!!)

The example script execution presumes that the maze is specified in the file input.txt. This may be changed if the maze is defined within the program.

If you are compiling your code under Cygwin shell, then you need to change line 5 to:

5. ./a.exe < input.txt

NOTE: Submissions that are not received correctly by the deadline will automatically receive a ZERO mark. **Late assignment submissions are not accepted!**

NOTES:

1. Your assignment must be RECEIVED by the due date and time. Late assignment submissions are NOT accepted. Keep your script file, and all your code unmodified as proof of its completion in case it is not received.

2. It is your responsibility to get an early start on the assignment, research and ask questions ahead of time from the due date.

3. Marks will be deducted for unclear code (improper spacing and alignment, hard to read programs and missing outputs).

4. Make sure you turn in a complete script file that clearly shows: your code, your compilation process, a listing of the directory showing your source file(s) and the a.out with the date/time stamps, and the output. DO NOT SUBMIT a.out FILES!

5. **PLAGIARISM:** CHEATING IS NOT TOLERATED – PLAGIARISM IS CHEATING! You must submit your own work. Students who are suspected of copying someone else's work will be reported to the department's chair and the Dean of Science and be dealt with in accordance with the University policies. You should not share your code with others. Codes that are similar to each other will BOTH be reported as potential evidence of copying. It is imperative that you write your own code.

6. Authorized/limited help on this assignment may be provided directly from your Lecture or Lab instructors and Teaching Assistants.