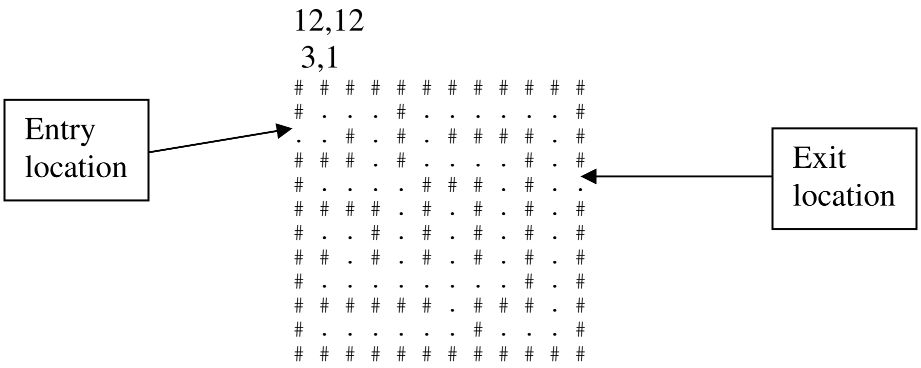
1. Write a function to traverse a maze.
2. The maze path is written in text file. The ‘#’ represent the wall and the ‘.’ Represent the path. The format of the input file is as follows:



```
12,12
 3,1
# # # # # # # # # # # #
# . . . # . . . . . . #
. . # . # . # # # # . #
# # # . # . . . . # . #
# . . . . # # # . # . .
# # # # . # . # . # . #
# . . # . # . # . # . #
# # . # . # . # . # . #
# . . . . . . . . # . #
# # # # # # . # # # . #
# . . . . . . # . . . #
# # # # # # # # # # # #
```

a. The first line represents the size of the maze viz. 12,12.
b. The second line gives the starting loacation of the maze viz. 3,1.
c. From the third line the maze starts.

3. In your program, read this input file.
4. Then write a function to traverse the maze.
5. You can use ‘Arrays’ or ‘Vectors’ to store the maze locations in your program.
6. If you are using the ‘Arrays’ to store the maze locations, then you will create a 12*12 array in your program and read only the text file which has 12*12 maze size. You can ignore the first line of the input text file in this case.
7. If you are using the Vectors, then you need to read the first line of the input text file and create a vector according to the size of the maze given in the file.
8. Keep in mind the following points when writing the traverse function:

There is a simple algorithm for walking through a maze which guarantees that you will find the exit. Suppose you 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, you will arrive at the exit of the maze eventually. There may be a shorter path than the one you have taken, but you are guaranteed to get out of the maze if you follow the algorithm.

9. **Hints**:

• The mazeTraverse() function should be passed the array/vector representing the maze, two starting coordinates (x and y) and the “right” wall.
• Write functions to determine if the coordinates are on the edge of the maze (only the starting and ending points are in this position) and if a move is legal. Also include a function to display the maze.

10. Sample output:

```
# # # # # # # # # # # #
# . . . # . . . . . . #
x . # . # . # # # # . #
# # # . # . . . . # . #
# . . . . # # # . # . .
# # # # . # . # . # . #
# . . # . # . # . # . #
# # . # . # . # . # . #
# . . . . . . . . # . #
# # # # # # . # # # . #
# . . . . . . # . . . #
# # # # # # # # # # # #
Hit return to see next move
...
# # # # # # # # # # # #
# 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 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 x x x x # x x x #
# # # # # # # # # # # #
Hit return to see next move
Maze successfully exited!
```