**Searching and Sorting: Recursion Drills - Maze**

In this challenge you will solve a problems using recursion. The objective is to practice using recursion to think about and solve problems. You do not have to worry about the efficiency of your solutions.

**Existing Files**

| **File** | **Description** |
| --- | --- |
| \_\_tests\_\_/maze.test.js | Contains tests for the maze problem. You are welcome to look at this file but do not modify it. |
| src/maze.js | Write your solutions to the maze problem in this file |

**Find a way out of the maze**

You have entered a Maze and need to find your way out of it. There are more than one possible paths through the Maze to the single exit point. Write a recursive function that will help you find a possible path through the maze.

You can use the following mazes to test your program.

let mySmallMaze = [

[' ', ' ', ' '],

[' ', '\*', ' '],

[' ', ' ', 'e']

];

let maze = [

[' ', ' ', ' ', '\*', ' ', ' ', ' '],

['\*', '\*', ' ', '\*', ' ', '\*', ' '],

[' ', ' ', ' ', ' ', ' ', ' ', ' '],

[' ', '\*', '\*', '\*', '\*', '\*', ' '],

[' ', ' ', ' ', ' ', ' ', ' ', 'e']

];

The Maze is represented as a N x M matrix (in the above case, a 3x3 or a 5x7 array). The starting point is the top left corner and the exit is indicated by e. You can't go outside the boundaries of the maze. The maze has passages that are blocked and you can't go through them. These blocked passages are indicated by \*. Passing through a blocked cell as well as passing though a cell that you have already passed before are forbidden.

For the above maze, a possible exit path can be RRDDLLDDRRRRRR.

Write a function that accepts a 2 dimensional array for teh maze and an array with 2 values for the current position in the maze. For example, you may call the function like this: mazeSolver(maze, [0, 0]) to indicate that the starting position is at the top left hand corner.