A maze is given as an n\*n binary matrix of blocks. The rat begins in the upper left most block (i.e maze[0][0]), and its destination block is the lower rightmost block i.e maze[n-1, n-1]. A rat starts from the source and has to reach the destination. The rat can only move forward and down.

In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination.

Ex:

{1, 0, 0, 0}

{1, 1, 0, 1} -> Maze has a path

{0, 1, 0, 0}

{1, 1, 1, 1}

{1, 0, 0, 1}

{0, 0, 1, 1} -> Maze does not have a path

{1, 1, 1, 1}

{1, 1, 1, 1}

Create a method that will return true if there is a valid path, and false if there is not. If a valid solution exists, print out the path it will take (hint: pass in an auxillary int[][] into the helper method to use to track the solution path).

a.) What are some edge cases to check for up front?

b.) What errors should you handle when you recurse?

c.) When should you stop recursing?

Please implement your solution to the problem below:

public boolean solveMaze(int[][] maze){