Recursion is the perfect tool for solving a maze. If you think about pathing through a maze, at each point you look at the paths you can take. If you always take the same order of paths (I will always go north first, then west, then south, then east) you will eventually find your way to the exit. For this lab, you are to implement a recursive method that will solve an ascii maze from the text files provided.

You have been given maze.java. This file already has the code necessary to read the file and store the maze as a 2D array. The characters mean the following:

* ‘S’ – starting point
* ‘G’ – ending point
* ‘.’ – pathing through the maze
* ‘#’ – wall of the maze
* ‘X’ – “Breadcrumbs” you mark as you traverse through the maze

There are two methods you must implement. The method headers are already defined for you. The **SolveMaze** method is called from the main and takes the maze (the 2D char array). This method is the helper method. This method must determine the starting coordinates of the maze by finding the ‘S’ character in the array. Once it has determined this, it must called **FindPath** and pass it the maze and the starting points. It will get a Boolean from **FindPath** and return it. It will be true or false based on whether or not it was able to find the path.

The **FindPath** method is the recursive method that will move through the maze. The algorithm for finding the path is outlined below:

1. If your x or y coordinates fall outside of the array, you will return false
2. If the path you are on is a wall or previously visited place you will return false
3. If the position is the goal (the ending point) you will return true
4. Recursivly find the path (note, the order you check the directions in does not matter, as long as you are checking for all 4 directions)
   1. If **FindPath**-north is true, return true
   2. If **FindPath**-west is true, return true
   3. If **FindPath**-south is true, return true
   4. If **FindPath**-east is true, return true
5. Lastly, if nothing else has been found, then that means the maze has no end point and the method should return false at the end

Two small things you must also implement in the **FindPath** method are marking your breadcrumbs. If you are in the position, you should mark it with an ‘X’ to signify you have visited the space. Also, if you are backtracking, you should remove the X and replace it with a ‘ . ‘

Lastly, it is nice to watch the maze get solved in a small form of animation. I have given you a **print** method in the code as well. In your **FindPath** method, everytime you update your path, you should call the **print** method passing it the maze.