Winner, Winner!

The following class is used to test board games. Many board games have a start and an end and some type of path or track you have to follow. To win the game, the players have to get from the start to the end and usually the faster the better.

public class Board

{

    private char[][] board;

private int maxPath; // length of the longest possible path

public Board(int rows, int cols, String line)

{ /\* to be implemented in part (a) \*/ }

/\*\* returns the length of the longest possible path in the board \*/

    public int getMaxPath()

{ /\* code not shown \*/ }

/\*\* calculates and returns the shortest path from S to X, if it exists.

\* if a path does not exist, return longest possible path or greater.

\* @param r is the row of "S"  
 \* @param c is the column of "S"

\*/

public int check(int r, int c )

{ /\* to be implemented in part (b) \*/ }

/\*\* precondition: S and X exist in board

\* postcondition: returns either the length of the path from S to X,

\* or -1, if no path exists.   
 \*/

public int win()

{ /\* to be implemented in part (c) \*/ }

}

(a)  Write the Board constructor method. The Board constructor will receive the number of rows, the number of columns, and a string of characters to be placed on the board.

For example, if the constructor call is new Board(3,3,"W-S-----X"), then the result would be a board of size 3x3 that looked as follows:

|  |  |  |
| --- | --- | --- |
| W | - | S |
| - | - | - |
| - | - | X |

  where W represents a wall, - represents a path, S represents the starting location, and X represents the exit.

Note that the length of the longest possible path for this board is 9.

Complete the Board constructor below.

public Board(int rows, int cols, String line)

{

(b) Write the Board method check. check will use **recursion** to see if this board is winnable or not winnable. A winnable board only occurs if a path of ‘-’S (dashes) leads from S to the X. The ‘-’ can only be connected up, down, left, and right. There are no diagonal connections. If the board is winnable, check will return the length of the shortest path from S to X. If the board is not winnable, check will return a number greater than or equal to the longest possible path for the given board. check will always start checking at S, the coordinates of which are passed to check.

**Example board 1**

- W - - - - -  
- W - W - W X

- - S - - - -

W - - - - W -

W - - W - - -game is winnable; shortest path is 5.

**Example board 2**

- W X -  
- W - W  
- W W -

S - - -

not winnable;

returns 16 (or greater).

In writing check, you may call any of the public methods of the Board class, including what you wrote in part (a).  Assume all of these methods work as specified regardless what you wrote in part (a).

/\*\* calculates and returns the shortest path from S to X, if it exists.

\* if a path does not exist, return longest possible path or greater.

\* @param r is the row of "S"  
 \* @param c is the column of "S"

\*/

public int check(int r, int c)

{

(c) Write the Board method win. The win method will search for S. When win finds S, it will return the length of the shortest path to X, if a path exists. If no such path exists, win will return -1.

In writing win, you may call any of the public or private methods of the Board class, including what you wrote in parts (a) and (b).  Assume that all of these methods work as specified regardless what you wrote in parts (a) and (b).

/\*\* precondition: S and X exist in board

\* postcondition: returns either the length of the path

\* from S to X, or -1, if no path exists.

\*/

public int win()

{