Programming Project #7
CMSC256 Fall 2012
Traversing a Maze

1 Project Description

Given the starting point in a 10 row × 20 column maze, your program is to discover and report if there is a path out of the maze. A maze consists of only hedges, footpaths, and a single exit. The program asks the user for the starting position in the maze (you may start at *any* point on a footpath except the exit, and the upper left corner is considered row 1, column 1) and then reports if they are free (i.e. you have found a way out) or trapped.

You may move vertically or horizontally (not diagonally!) in any direction in the maze as long as you are on a footpath or at the exit; you may not move into (onto) a hedge. If you move into the exit square, you have successfully exited the maze. If you have tried all of the possible paths out of the maze without success, you are trapped. *You must use a recursive routine to search for the exit.* If you are in a square and no progress is possible (i.e. the squares surrounding you either are blocked with hedges and/or you have already tried those paths), you must 'go back the way you came' and try another path (this will be handled 'automatically' by the recursive algorithm). The path you find out of the maze, if one exists, need not be the *shortest* path.

2 Input

Input is a 10 row by 20 column array of characters (containing only `1`s, `0`s, a single `E`, with no spaces) from an text data file specified on the command line; you may assume the maze data file is correct. Each row in the data file corresponds to a row in your maze. Hedges are indicated by the character `1`, paths by the character `0`, and the exit is marked by an upper case `E`. The starting point (i.e. a row, column pair) in the maze will be input from the keyboard; you should make sure it is a valid position (see the sample run).

Here is an example data file:

```
00001110001000000100
11100011101010001111
11111000100000111000
E0011110101100010010
01011010101111000110
00000000100110011110
11011100110110111000
00011110110111111101
01011011110110000001
01000000000110110111
```

3 Output

You should format your output *exactly* as the sample run shown below; also use *exactly* the same labels, spacing, data prompts, and error messages. Echo print the (initial) maze prior to asking the user for their starting point. After acquiring the start position, print the complete maze with an S in the starting point followed by the words 'I am free' if you have found a path out of the maze or the words 'Help, I am trapped' if you cannot.

Your maze should be printed using one screen column and screen row per maze column and row using exactly the same characters as shown in the sample run. You should print a border around the maze and label *all* sides with the appropriate row and column number.

4 Sample Run

Below is a sample run using the example data file given above.

```
liberty:~/javaprogs/% java RMaze datafile
Maze Traversal - D. Resler 11/09
            1         2
   12345678901234567890
  +--------------------+
 1|    ###   #      #  |1
 2|###   ### # #   ####|2
 3|#####   #     ###   |3
 4|E  #### # ##   #  # |4
 5| # ## # # ####   ## |5
 6|        #  ##  #### |6
 7|## ###  ## ## ###   |7
 8|   #### ## ####### #|8
 9| # ## #### ##      #|9
10| #         ## ## ###|10
  +--------------------+
            1         2
   12345678901234567890
Starting point? row: -3
                col: 5
Illegal starting position; try again
Starting point? row: 4
                col: 1
Try again - you cannot start at the exit
Starting point? row: 2
                col: 12

            1         2
   12345678901234567890
  +--------------------+
 1|    ###   #      #  |1
 2|###   ### #S#   ####|2
 3|#####   #     ###   |3
 4|E  #### # ##   #  # |4
```

```
 5| # ## # # ####   ## |5
 6|        #  ##  #### |6
 7|## ###  ## ## ###   |7
 8|   #### ## ####### #|8
 9| # ## #### ##      #|9
10| #         ## ## ###|10
  +--------------------+
           1         2
   12345678901234567890
I am free!
liberty:~/javaprogs/%
liberty:~/javaprogs/% java RMaze datafile
Maze Traversal - D. Resler 11/09
           1         2
   12345678901234567890
  +--------------------+
 1|    ###   #      #  |1
 2|###   ### # #   ####|2
 3|#####   #     ###   |3
 4|E  #### # ##   #  # |4
 5| # ## # # ####   ## |5
 6|        #  ##  #### |6
 7|## ###  ## ## ###   |7
 8|   #### ## ####### #|8
 9| # ## #### ##      #|9
10| #         ## ## ###|10
  +--------------------+
           1         2
   12345678901234567890
Starting point? row: 1
                col: 20

           1         2
   12345678901234567890
  +--------------------+
 1|    ###   #      # S|1
 2|###   ### # #   ####|2
 3|#####   #     ###   |3
 4|E  #### # ##   #  # |4
 5| # ## # # ####   ## |5
 6|        #  ##  #### |6
 7|## ###  ## ## ###   |7
 8|   #### ## ####### #|8
 9| # ## #### ##      #|9
10| #         ## ## ###|10
  +--------------------+
           1         2
   12345678901234567890
Help, I am trapped!
liberty:~/javaprogs/%
```

m	[1]	[2]	[3]	[4]	[5]
[1]	E	1	0	0	1
[2]	0	0	0	1	1
[3]	0	1	0	1	1
[4]	0	1	0	1	0
[5]	1	1	1	1	0

Figure 1: Example 5 × 5 maze

5 Discussion

You are to find your way out of the maze by calling a recursive method (let's call it p) that is passed a legal starting point and the maze m and returns true if it's found a way out, false otherwise. Figure 1 shows a simple 5 × 5 maze that will be used in the following example. Suppose we start with an entry point of [3,1], i.e. we wish to determine if $p([3,1], m)$ returns true or false. How do we do this?

For any position [r,c] in the maze there are a (maximum) of 4 possible moves:

	$[r-1, c]$	
$[r, c-1]$	**[r,c]**	$[r, c+1]$
	$[r+1, c]$	

An exhaustive search of the maze will require that we choose each of these moves in turn (using an arbitrary order) until we find one that leads us out of the maze. So in our example we might try the following moves out of [3,1]: [2,1], [3,2], 4,1], [3,0]. But notice that moves [3,2] and [3,0] are illegal, leaving us with only two possibilities—if a path out of the maze exists, either $p([2,1], m)$ or $p([4,1], m)$ will return true.

Suppose we try $p([4,1], m)$ first. Obviously this leads to a dead end, but what are the legal moves out of position [4,1]? There is only one, the square that we just left: [3,1]. But since we're only interested in forward progress towards an exit, we don't wish to move back into a square we've already visited. How do we prevent this? We need to leave bread crumbs (i.e. markers) in every square that we've already visited. So now we choose to only move into squares that are legal moves *and* are squares that we have not visited before.

Here therefore is our general algorithm:

```
subroutine p([r,c],m) {
  if [r,c] is the exit then
     report success
  else if [r,c] is not a legal move then
     report failure
  else {
     mark [r,c] with a bread crumb
```

```
     if [r-1,c] is a way out then
        report success
     else if [r,c+1] is a way out then
        report success
     else if [r+1,c] is a way out then
        report success
     else if [r,c-1] is a way out then
        report success
     else
        report failure
  }
}
```

6 Extra Credit

For 15% extra credit, also print the path (by using a series of pluses (+)) you took through the maze should one be found. For example:

```
liberty:~/javaprogs/% java RMaze datafile
Maze Traversal - D. Resler 11/09
            1         2
   12345678901234567890
  +--------------------+
 1|    ###   #      #  |1
 2|###   ### # #   ####|2
 3|#####   #     ###   |3
 4|E  #### # ##   #  # |4
 5| # ## # # ####   ## |5
 6|        #  ##  #### |6
 7|## ###  ## ## ###   |7
 8|   #### ## ####### #|8
 9| # ## #### ##      #|9
10| #         ## ## ###|10
  +--------------------+
            1         2
   12345678901234567890
Starting point? row: 4
                col: 10
```

```
           1         2
   12345678901234567890
  +--------------------+
 1|    ###   #      #  |1
 2|###   ### # #   ####|2
 3|#####   #     ###   |3
 4|E++#### #S##   #  # |4
 5| #+## # #+####   ## |5
 6|  +     #++##  #### |6
 7|##+###  ##+## ###   |7
 8|  +#### ##+######## #|8
 9| #+## ####+##      #|9
10| #++++++++## ## ###|10
  +--------------------+
           1         2
   12345678901234567890
I am free!
liberty:~/javaprogs/%
```

Note that the path shown need not be the *shortest* path. Extra credit will only be given if you successfully (correctly) print out the complete path (i.e. no 'partial' extra credit will be given). If you have completed work for extra credit, please clearly indicate this in the comment section at the top of the listing for your program.

7 Deliverables

Name your source file `Proj7`*XXXX*`.java` where *XXXX* is the last 4 digits of your student id number, and submit it in the usual way via the class web page.

Due date: Monday December 3^{rd}