DESIGN PROBLEM–I

Date of Allocation: 2nd Mar, 09
Date of Submission: 9th Mar, 09
Mode of Submission: Online Only (through UMS)

Escape from Maze

LPU has decided to create a maze, next to it's library, so that only intelligent people can go into the library. As is typical of mazes, this one too has several dead ends and only one correct start and end point. You have to write a program to navigate this maze and enter the library. Your starting point in the maze is marked by the character '#' and the entrance to the library (i.e., end point of the maze) is marked by the character '@', i.e. your program must start from '#' and end at '@'.

The maze is composed of a rectangular array of cells. Apart from the start (#) and end (@) cells, maze contains solid cells and empty cells. A valid path never passes via a solid cell. The solid cells are marked by 0s and empty cells are marked by 1s. Once on a cell, you can only move to any one of the neighbouring four (i.e., north, east, west, south) cells.

Note:Assume that no path will ever cross itself i.e there will be no loops in any path.

Input specification:

- The first line of input will be two integers, R and C, specifying the number of rows and columns.
- The next R lines contain C characters each separated by white space, thereby specifying the layout of the maze. It is guaranteed that the maze will contain only one starting and one ending character. Also, it is guaranteed that the maze will contain only one correct path, from the start point to the end point.

Output specification:

- Your program must output on one line, the number of hops required to reach the end point.

Example:
Input:
5 5
0 0 0 0 0
0 0 0 0 0
0 # @ 0 0
0 0 0 0 0
0 0 0 0 0
Output:
1
Input:
6 6
0 0 0 0 @ 0
0 1 0 1 1 0
1 1 1 1 0 1
0 1 0 1 0 0
0 0 # 1 1 1
0 1 1 0 0 1
Output:
6

Solution

```
import ncst.pgdst.*;
class Maze
{
	int m,n,sr,sc;
	boolean found = false;
	char maze[][];
	Maze(int tm,int tn)
	{
		m = tm; n = tn;
		maze = new char[m][n];
	}
	public static void main(String args[]) throws Exception
	{
		SimpleInput in = new SimpleInput();
		Maze mm = new Maze(in.readInt(),in.readInt());
		mm.readInput(in);
		mm.print();
		mm.process(mm.sr,mm.sc,1,0);
	}
	void readInput(SimpleInput in) throws Exception
	{
		for(int i=0; i < m ;i++)
			for(int j=0; j < n ;j++)
			{
				in.skipWhite();
				maze[i][j] = in.readChar();
				if(maze[i][j] == '#')
				{sr = i; sc = j;}
			}
	}
	void process(int cr,int cc,int di,int hop)
	{
		//System.out.println(maze[cr][cc] + " " + cr + " " + cc + " " + m + " " + n);
		if(cr >= 0 && cr < m && cc >= 0 && cc < n)
		{
			if(maze[cr][cc] == '@')
			{	found = true;
				System.out.println("Here is the hops : " + hop);}
			if(maze[cr][cc] != '1' && maze[cr][cc] != '#')
				return;
			else
				maze[cr][cc] = 0;
			while(found == false)
			{
			System.out.print(maze[cr][cc] + " " + di + " ");
				if(di == 1){
					process(cr-1,cc,di,hop+1);
					//System.out.print("returns ");
					di = 3;
					//System.out.print("heres di : " + di);
				}
				else if(di == 3){
					process(cr+1,cc,di,hop+1);
					di = 2;
				}
				else if(di == 2){
					process(cr,cc+1,di,hop+1);
					di = 4;
				}
				else if(di == 4){
					process(cr,cc-1,di,hop+1);
					di = 1;
				}
			}
```

```
            }
        }
        void print()
        {
            for(int i=0; i < m ;i++)
            {
                for(int j=0 ; j < n ;j++)
                    System.out.print(maze[i][j] + " ");
                System.out.println();
            }
        }

}
```