ECOO 2014

Programming Contest Questions

Regional Competition (Round 2)

April 26, 2014

Sheridan | Get Creative

Questions made possible in part through the support of the
Sheridan College Faculty of Applied Science and Technology.

Problem 1: Scratch and Win

Play the “MOE Millions” scratch and win card and you could win up to $1 000 000! Or you could win less than that! Or you might win nothing at all. There are 10 possible prizes: $1, $2, $5, $10, $50, $100, $1000, $10 000, $500 000 and $1 000 000.

To play, scratch off the squares on a 3x3 grid one at a time. If you find 3 matching prize amounts under the scratchy stuff, you win that amount. It’s that simple! Each card will contain a maximum of one set of 3 matching symbols. Employees of the Ministry of Education and their families are not eligible for any prizes.

$10	$100	
$10	$1	$50
$50	$1000	$1

OK, you’ve got your card, and you’ve scratched off 8 of the squares as shown on the right. What fabulous prizes could you win when you scratch off that final square?

DATA11.txt (DATA12.txt for the second try) will contain 10 test cases. Each test case will consist of nine lines representing the nine squares on the card. The first line is for the top left box, the second is for the top middle box, then top right, then middle row left, and so on down to the bottom right corner. If a box has been scratched, the line for that box will contain the prize amount that is revealed. If not, the line will contain a question mark. The cards could be in any state of play, ranging from just starting out (no boxes scratched yet) to completely finished (all boxes scratched).

Your job is to output a list of all prizes the cardholder can or will win in order from lowest to highest, separated by spaces. Each card in the input should be represented by a single line in your output. If no prize is possible, output the exact string “No Prizes Possible”.

Note that the sample input below only contains one test case, but the real data files will contain 10 test cases, one after another, with no blank lines between. Your output should therefore consist of 10 lines.

Sample Input

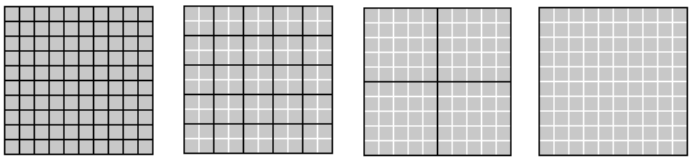
```
$10
$100
?
$10
$1
$50
$50
$1000
$1
```

Sample Output

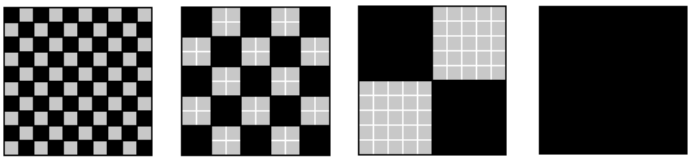
```
$1 $10 $50
```

Problem 2: Black and Grey

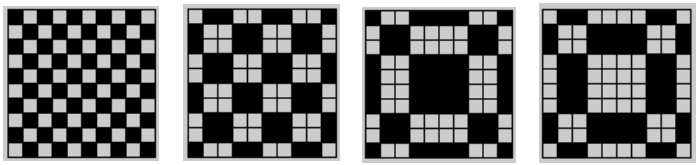
You have a 10x10 board made of tiles that are black on one side and light grey on the other, but right now they're all showing their grey side. Imagine the patterns you could make! For example, suppose you imposed a grid over the tiles to divide them into squares of the same size with no tiles left over. There are four grid sizes that would work: 1x1, 2x2, 5x5 and 10x10, as shown below (the tiles are in grey, the grids are shown with black lines).



Now imagine that each grid is a checkerboard where the top left square is a black square, like this:



Now imagine flipping over all the tiles that are underneath black grid squares. Do the 1x1 grid first, then 2x2, then 5x5, then 10x10. You're going to get some cool patterns. The pictures below show the results.



DATA21.txt (DATA22.txt for the second try) will contain 10 test cases. Each test case consists of six lines. The first line contains a single integer N ($1 \le N \le 1000000$) representing the size of the board. The next 5 lines each contain two integers R and C, separated by a space. R and C represent the row and column of a tile on the board ($1 \le R,C \le N$). Rows are numbered top down starting at 1 and columns are numbered left to right starting at 1. Your job is to simulate the pattern-making process described above for an NxN board, and then output a single line of 5 characters representing the five tiles in the locations given in the test case, in the order they originally appeared in the file. Each tile should be represented by an uppercase B or G depending on whether it is showing its black or grey side in the final pattern. Note that there are 2 test cases in the sample data below, but the real data files will contain 10 test cases.

Sample Input

```
10
5 1
5 2
5 3
5 4
5 5
12
6 6
7 7
8 8
9 9
10 10
```

Sample Output

```
GBBGG
GGGGG
```

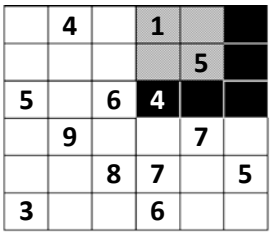
Problem 3: EasySweeper

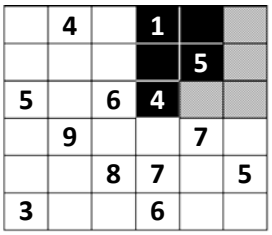
There's a popular puzzle game where you try to uncover mines on a grid. EasySweeper is a version of this game, but it's a little different.

	4		1		
				5	
5		6	4		
	9			7	
		8	7		5
3			6		

EasySweeper is played on a grid, and each grid square either contains a mine or it doesn't. When you start you get a partially filled-in grid of integer clues like the one shown at right. Think of a clue as being at the center of a 3x3 area. The clue itself represents the total number of mines in that 3x3 area.

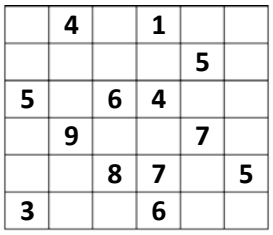
The pictures below show two of the 126 possible arrangements of mines represented by the "5" in the top right corner of the example (a black square represents a grid space with a mine in it and a shaded square represents an empty grid space). Only one of the 126 possible arrangements will work when all the other clues are taken into account (in fact, the second one shown below can already be ruled out because of the "1" clue above and to the left of the "5"). Your task is to figure out an arrangement of mines that satisfies all the clues on the board at once.



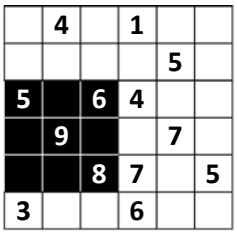


This game is called EasySweeper because solving the puzzles will never require any advanced logic or guesswork.

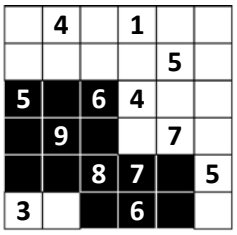
For example in the puzzle shown, there must be mines in all the grid squares around the '9' and around the '6' on the bottom row. Once you've marked those squares as containing a mine, you can see that the '7' just above the '6' you filled in is satisfied – you have found all 7 mines. That means there are no mines in the remaining two places. You can continue like that until you find the unique solution to the puzzle. This process is diagrammed below.



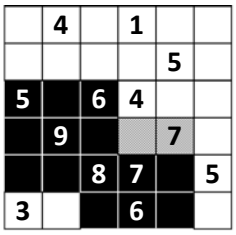
the starting board…



fill in around the '9'…



fill in around the '6'…



the '7' is done…

DATA31.txt (DATA32.txt for the second try) will contain 5 test cases. Each test case consists of N lines of N characters, representing an NxN puzzle grid ($5 \le N \le 20$). Each character will either be a clue from 0 to 9 or it will be a hyphen character (ASCII code 45). Your job is to find the unique solution for the puzzle and print it in the format shown below, where an uppercase 'M' stands for a mine and a dot character for no mine. You should separate each board in your output file by a blank line, but there will be no blank lines separating boards in the input file.

To help the judges, you should configure your IDE so that its text output is displayed in a fixed-width font like "courier new". If you can't do that, you can cut and paste the output into notepad or a similar text editor that displays in a fixed-width font. Note that there are only two test cases in the sample data, but the real data files will contain 5 test cases each.

Sample Input

```
-4-1--
----5-
5-64--
-9--7-
--87-5
3--6--
2-20---33-
3-2---3---
-2----222-
-210---3-3
2-1--6-5-4
23--44---4
---5-42---
1-5--1-02-
----2-1--3
0-1-01-1--
```

Sample Output

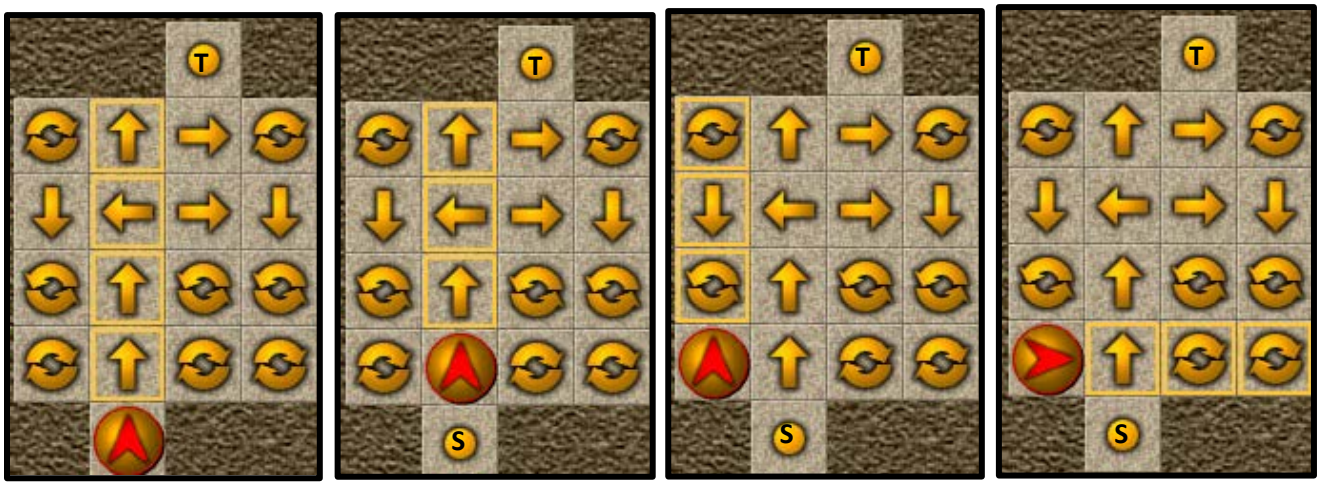
```
MMM..M
.M...M
MMMMMM
MMM..M
MMMMMM
.MMMMM

.M...MMM.M
.M......M.
M........M
.....MM...
.M...M.MMM
M...MMM.MM
..MM......
.M.MM....M
..M......M
......M..M
```

Problem 4: What Lies Ahead

You are trapped in a room with a puzzle grid painted on the floor. You are in a start square, just below the puzzle grid, facing up. Your task is to get to a target square (or suffer a terrible fate). Your first move must be to step into the main puzzle grid if you can. Once in, you may not step outside the grid unless it is onto a target square. Once you step onto a target square, you are finished and can make no further moves.

Each square in the puzzle grid has a symbol painted on it representing a possible move (up, down, left, right, clockwise, or counterclockwise). You can only move one grid square at a time and you can only turn 90 degrees at a time. Your next move must always be one of the moves you can see painted on the floor ahead of you (this does not include the one you are standing on). You cannot change the direction you are facing unless you can see a move ahead that allows you to turn clockwise or counter-clockwise. Because of the restrictions on the direction you can face, you will often end up stepping sideways or backwards when completing this puzzle.



In the sequence of moves shown above, the user starts at the bottom, facing up. Her target is the square above the maze at the top. For her first move, the choices are UP or LEFT (just the moves she sees in front of her). She can't go left, so she chooses UP. Then her choices are UP or LEFT again and she chooses LEFT. Now her choices have changed: CLOCKWISE, COUNTERCLOCKWISE, or DOWN. She can't move down and if she turns counterclockwise, she will have no moves in front of her. So she chooses CLOCKWISE. For her next move, she can now choose from UP or CLOCKWISE.

DATA41.txt (DATA42.txt for the second try) will contain 10 test cases. Each test case consists of 6 lines of 6 characters, representing a 4x4 puzzle grid as well as all possible start and target squares around the outside of the grid. The moves are U, D, L, R, C, and B for UP, DOWN, LEFT, RIGHT, CLOCKWISE, and COUNTERCLOCKWISE respectively. The start square is marked S and can be anywhere on the bottom row. There will be five possible target squares, marked T, which can appear anywhere around the

outside of the puzzle grid. Squares which are out of bounds are marked with the "." character (ASCII code 46).

Write a program to determine which of the target squares can be reached from the given start square. You should output a single line of 5 characters, with each character representing a target square (in order from top left, scanning each row from left to right, moving downwards). If a target square is reachable, output T for that square. If not, output F.

Note that in the sample input below, there are 2 test cases, but the real data files will contain 10. The squares which are part of the puzzle grid are shaded below for ease of reading, but this shading will not appear in the file.

Sample Input

```
..TT..
TCURC.
.DLRD.
.BUBB.
.CUCCT
.TS...
.T.T..
TRCDU.
.UCLDT
.RCBL.
TDUCU.
.S....
```

Sample Output

```
TTFFT
FTFFF
```

Puzzle Concept:	Andrea Gilbert
Puzzle Boards:	James Stephens (original boards have been adapted for ECOO)
Puzzle Images:	www.clickmazes.com

Question Development Team	**President of ECOO-CS**
Sam Scott (Sheridan College)	David Stermole
Kevin Forest (Sheridan College)	
Greg Reid (St. Francis Xavier Secondary School, Mississauga)	**Communications**
Dino Baron (University of Waterloo)	John Ketelaars
Nathan Schucher (University of King's College)	