31st Annual Rowan University Programming Contest

hosted by the
Computer Science Department

Friday, 7 April 2017

Contest Problem

1 Introduction

We want to build a robot that navigates hazards. As a preliminary step, we're going to build a simulator in software.

Our surface will be read as a two-dimensional array of numbers, with values indicating clear territory, obstacles that can be changed (such as a boulder that might be moved out of the way), and obstacles that cannot be changed (such as a hole that we can't fill in).

Write a program to read in the numbers representing the environment, the robot's initial location and abilities, and the movement instructions, and then print out the results of the robot's movements.

2 Detailed Description

2.1 Simulated Environment

The simulated environment will be a two-dimensional array of integers, all between -9 and 9, inclusive. These numbers represent square regions on a flat table. For example, this grid:

```
0  1  0  0  1  0  2
0  0  0  0 -2 -4  0
1  0 -3  0  0  1  1
3  1  5  0  1  0  6
```

describes a 7×4 table with several obstacles, represented by the nonzero values. The positive numbers represent obstacles which can be altered by a robot not strong enough to get over them in one step. The negative numbers represent obstacles that cannot be altered, but which might be manageable as they are.

The coordinate grid for all tables will begin with (0, 0) in the upper-left corner, with positive X values to the right and positive Y values going down. When listing coordinates, the X value always comes first.

For the table above, the -2 is at coordinate (4,1), the 5 is at coordinate (2,3), the 3 is at coordinate (0,3), and the 6 is at coordinate (6,3).

2.2 Robot Abilities

The robot will get a setting for *strength*, which corresponds to its ability to change or manage obstacles represented by positive numbers.

The robot will get a setting for *agility*, which corresponds to its ability to maneuver around obstacles represented by negative numbers.

2.3 Movement Instructions

The robot's initial position will be given as a coordinate on the table.

Movement instructions will be a series of letters from the set {N, S, E, W}, indicating, respectively, *north*, *south*, *east*, and *west*.

For this problem, north means toward a lower Y value, south means toward a higher Y value, east means toward a higher X value, and west means toward lower X value.

If the initial position is not on the table at all, you should print a message that the input was invalid and go on to the next simulation.

If the initial position is on a positive number, change it to a zero immediately and proceed as if the robot was placed on a zero.

If the initial position is on a negative number whose absolute value is greater than the robot's agility setting, you should print that the robot fell in a hole before starting and go on to the next simulation.

2.4 Robot Actions

If the robot is instructed to move onto a spot which has a zero, or a positive number less than the robot's strength, it should move onto the square. (This represents a small obstacle.) If instructed to move onto a spot which has a positive number greater than the robot's strength, it should not move, but the number on the target square should be reduced by the robot's strength value. (This represents the robot slowly managing difficult terrain.) If instructed to move onto a spot which has a positive number equal to the robot's strength, the number in that square should be set to zero and the robot should move onto that spot. (This represents a larger obstacle that can still be overcome in one move.) If instructed to move onto a spot with a negative value which has an absolute value less than or equal to the robot's agility, the robot should move onto that square. (This represents a hole the robot can safely cross.) If it is instructed to move onto a spot with a negative value which has a higher absolute value than the robot's agility, the robot should do nothing. (This represents a hole too big to cross.)

If instructed to move off the table, it should print a warning message and stop processing instructions.

3 Example Simulation #1

A robot (location indicated using square brackets) which has strength 3 and agility 2 has been placed at (0,0), with instructions E S S E S E E.

The start state looks like this:

```
[0] 1  0  0  1  0  2
 0  0  0  0 -2 -4  0
 1  0 -3  0  0  1  1
 3  1  5  0  1  0  6
```

'E' means to move right. The robot's strength is higher than the obstacle's value, so it moves.

```
 0 [1] 0  0  1  0  2
 0  0  0  0 -2 -4  0
 1  0 -3  0  0  1  1
 3  1  5  0  1  0  6
```

'S' means to move down. Because the square has value 0, it just moves to that spot.

```
 0  1  0  0  1  0  2
 0 [0] 0  0 -2 -4  0
 1  0 -3  0  0  1  1
 3  1  5  0  1  0  6
```

'S' means to move down.

```
 0  1  0  0  1  0  2
 0  0  0  0 -2 -4  0
 1 [0]-3  0  0  1  1
 3  1  5  0  1  0  6
```

'E' means to move right. The robot's agility is 2, and the obstacle is -3, so the robot does nothing.

```
 0  1  0  0  1  0  2
 0  0  0  0 -2 -4  0
 1 [0]-3  0  0  1  1
 3  1  5  0  1  0  6
```

'S' tells the robot to move down.

```
 0  1  0  0  1  0  2
 0  0  0  0 -2 -4  0
 1  0 -3  0  0  1  1
 3 [1] 5  0  1  0  6
```

'E' tells the robot to move right. The robot cannot move there. It reduces the obstacle by 3.

```
 0  1  0  0  1  0  2
 0  0  0  0 -2 -4  0
 1  0 -3  0  0  1  1
 3 [1] 2  0  1  0  6
```

'E' means to move right. The robot's strength is greater than the obstacle's value, so it moves.

```
 0  1  0  0  1  0  2
 0  0  0  0 -2 -4  0
 1  0 -3  0  0  1  1
 3  1 [2] 0  1  0  6
```

(This simulation corresponds to the Sample Input #1 on page 4.)

4 Input

4.1 Input Specification

For text input, your program should accept input in the following format:

1. An integer, $\mathcal{D}$, where $1 \leq \mathcal{D} \leq 100$, which is the number of datasets in this file.
2. $\mathcal{D}$ data sets, each of which is in this format:
   (a) One line with two integers, $\mathcal{W}$ and $\mathcal{H}$, where $1 \leq \mathcal{W} \leq 50$, and $1 \leq \mathcal{H} \leq 50$, indicating the width and height of the grid representing the table.
   (b) $\mathcal{H}$ lines, each with exactly $\mathcal{W}$ integers separated by spaces, which are the initial values of the squares on the table.
   (c) One line with two integers, $\mathcal{X}$ and $\mathcal{Y}$, indicating the robot's start position.
   (d) One line with two integers, $\mathcal{S}$ and $\mathcal{A}$, indicating the robot's strength and agility.
   (e) One line with one integer, $\mathcal{M}$, where $1 \leq \mathcal{M} \leq 50$, which is the number of movement instructions the robot is to process.
   (f) One line with $\mathcal{M}$ letters, separated by spaces, representing directions the robot is to move. (As there are a maximum of 50 instructions, this line will have at most 49 spaces, for a total length of 99 characters.)

4.2 Sample Input #1

Data in file	Item #	Meaning in plain English
`1`	`1`	*this file has 1 data set*
`7 4`	`2.a`	*Data Set 1 has a 7x4 table*

`0 1 0 0 1 0 2`
`0 0 0 0 -2 -4 0`
`1 0 -3 0 0 1 1`
`3 1 5 0 1 0 6`
	`2.b`	*the layout of the table*

`0 0`	`2.c`	*the robot starts at the upper-left corner*
`3 2`	`2.d`	*the robot has strength 3 and agility 2*
`7`	`2.e`	*the robot will process 7 instructions*
`E S S E S E E`	`2.f`	*the 7 movement commands*

(This input, which corresponds to the example simulation on page 3, is on the website as **sample1.txt**.)

You may choose to have your program read the input from the keyboard, or ask the user for a filename and then read the file. Users of GUI-based programming environments may prefer to use text boxes into which the values can be entered, and buttons to begin their calculation. Any reasonable variation in the spirit of the problem is acceptable.

You need not do error-checking on the input. Each line will have exactly the number of items described with no stray characters or extra spaces. There will be no blank lines.

All sample and test data sets are available at <http://elvis.rowan.edu/rupc/2017>

5 Output

5.1 Output Specification

For each data set configuration, your program must generate output as follows:

1. The text '`Analyzing D data sets`', where D is the number of data sets in the input.
2. For each data set, print the following:
   (a) '`Data Set D`', where D is the number of the data set being reported on.
   (b) '`Robot Start: X Y`', where X and Y are the robot's start coordinates.
   (c) '`Robot Strength and Agility: S A`', where S and A are the robot's strength and agility.
   (d) '`Instructions: I`', where I is the number of instructions being carried out.
   (e) Depending on the input, one of the following results:
      i. The text '`robot is off table`', if the initial position is not within the coordinates of the grid. Go immediately to the next data set.
      ii. The text '`robot is in a hole`', if the initial position is on a negative number with greater absolute value than the robot's agility. Go immediately to the next data set.
      iii. The text '`Instruction N unsafe: C at X Y`', if the robot is instructed to roll off the table, where N is the number of the instruction in the sequence, C is the instruction given, and X, Y specify the robot's location when the unsafe instruction was received.
      iv. The text '`I instructions processed; robot at X Y`', if all instructions could be safely executed, where I is the number of instructions and X, Y are the robot's final coordinates.
   (f) For cases (e.iii) and (e.iv), print the text '`Table configuration:`', followed by the final values on the table. (You do **not** have to print square brackets showing the robot's location. That was only to make the simulations easier to follow.)

5.2 Sample Output #1

```
Analyzing 1 data set(s)
  Data set 1
     Robot Start: 0 0
     Robot Strength and Agility: 3 2
     Instructions: 7
     7 instructions processed; robot at 2 3
     Table configuration:
         0  1  0  0  1  0  2
         0  0  0  0 -2 -4  0
         1  0 -3  0  0  1  1
         3  1  2  0  1  0  6
```

(This output corresponds to Sample Input 1 from page 4.)

Your output does **not** have to duplicate the sample output exactly with respect to spacing or use of upper/lower case. Your output should be neat, but need not match character-for-character.

6 Example Simulation #2

A robot with strength 1 and agility 0 has been placed at (0,0) with instructions S S E N S S E E S S W N.

The start state looks like this:

```
[0]  2   0   0   1
 0   0   0   0   8
 2   0  -1   0   1
 3   1   5   0   1
 0   0   0   1   0
```

Command #1, 'S', tells the robot to move down.

```
 0   2   0   0   1
[0]  0   0   0   8
 2   0  -1   0   1
 3   1   5   0   1
 0   0   0   1   0
```

Command #2, 'S', means to move down. The robot cannot move, but the number is decremented:

```
 0   2   0   0   1
[0]  0   0   0   8
 1   0  -1   0   1
 3   1   5   0   1
 0   0   0   1   0
```

Command #3, 'E', tells the robot to move to the right.

```
 0   2   0   0   1
 0  [0]  0   0   8
 1   0  -1   0   1
 3   1   5   0   1
 0   0   0   1   0
```

Command #4, 'N', tells the robot to move up. The robot cannot move, but the number is decremented:

```
 0   1   0   0   1
 0  [0]  0   0   8
 1   0  -1   0   1
 3   1   5   0   1
 0   0   0   1   0
```

Command #5, 'S', tells the robot to move down.

```
 0   1   0   0   1
 0   0   0   0   8
 1  [0] -1   0   1
 3   1   5   0   1
 0   0   0   1   0
```

Command #6, 'S', tells the robot to move down.

```
 0   1   0   0   1
 0   0   0   0   8
 1   0  -1   0   1
 3  [0]  5   0   1
 0   0   0   1   0
```

Command #7, 'E', tells the robot to move right. The robot cannot move, but the number is decremented.

```
 0   1   0   0   1
 0   0   0   0   8
 1   0  -1   0   1
 3  [0]  4   0   1
 0   0   0   1   0
```

Command #8, 'E', tells the robot to move to the right.

```
 0   1   0   0   1
 0   0   0   0   8
 1   0  -1   0   1
 3  [0]  3   0   1
 0   0   0   1   0
```

Command #9, 'S', tells the robot to move down.

```
 0   1   0   0   1
 0   0   0   0   8
 1   0  -1   0   1
 3   0   3   0   1
 0  [0]  0   1   0
```

Command #10, 'S', tells the robot to move down. That would be off the table, so the robot stops processing instructions.

(This simulation corresponds to the first data set in Sample Input #2 on page 7.)

7 Sample #2

7.1 Sample Input #2

```
2
5 5
0 2 0 0 1
0 0 0 0 8
2 0 -1 0 1
3 1 5 0 1
0 0 0 1 0
0 0
1 1
12
S S E N S S E E S S W N
5 5
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
1 1
1 1
15
E E E E S S S S W W W W N N N
```

7.2 Sample Output #2

```
Analyzing 2 data set(s)
Data set 1
   Robot Start: 0 0
   Robot Strength and Agility: 1 1
   Instructions: 12
   Instruction 10 unsafe: S at 1 4
   Table configuration:
        0  1  0  0  1
        0  0  0  0  8
        1  0 -1  0  1
        3  0  3  0  1
        0  0  0  1  0
Data set 2
   Robot Start: 1 1
   Robot Strength and Agility: 1 1
   Instructions: 15
   15 instructions processed; robot at 1 1
   Table configuration:
        2  2  2  2  2
        2  0  0  0  2
        2  0  2  0  2
        2  0  0  0  2
        2  2  2  2  2
```

8 Test Data

The contest website, <http://elvis.rowan.edu/rupc/2017>, has three test data files: **test1.txt**, **test2.txt**, and **test3.txt**.

Run your program on those three files and print the results. **You must submit printed output to earn full points.** Your program will also be run on data known only to the judges.