CS1020 Sit-in Lab 04 A - Robot Navigation

Semester 2 AY2013/2014

Requirements

Use recursion to solve the tasks in this sit-in lab. NO for/while/do..while loops should be used in your code except for getting input. **No correctness marks will be given for iterative solutions. Correctness marks will be halved for code that uses for/while/do..while loops within the recursive solutions.**

Problem Description - Part 1 (4 test cases/28 marks)

You have joined an indoor robot navigation contest. For this contest, a huge hall is used as the terrain for the robot to move in. The robot will be placed at some starting point within the hall and it will have to move to a specified destination. You job is to write a program to help navigate the robot from the starting point to the destination. For part 1, the hall is partitioned into a one dimensional grid which is represented as a integer array of size N. An example where N = 7 is given below

3	7	2	1	12	20	8

Each array index represents a fixed sized cell within the hall. The integer value at each array index represents the "cost" that the robot will incur if it moves into that cell. **The robot can start from any cell (this is given in the input) and it must move towards the rightmost cell which is the destination**.

Write a recursive method to calculate the total cost incurred by the robot to move from the starting cell to the destination cell. This cost is inclusive of the cost of the starting cell. For example, if the robot starts from index 2 in the example above, the total cost is 2+1+12+20+8 = 43.

3	7	**2**	**1**	**12**	**20**	**8**

You can assume that the total cost of moving the robot from the starting cell to the destination cell **will not exceed 1000000**, and that the **value in each cell is unique**.

Input

The 1st line in the input is the value 1. This indicates the input is for part 1. The next line is an integer **N** (**N** $\geq$ 1) indicating the size of the array. The following line is the starting index of the robot. The next line contains N integers separated by a white space and they represent the cost for each cell. An example input is given below:

```
1  <-- indicates this is input for part 1
7  <-- size of the array
2  <-- starting index of robot
3 7 2 1 12 20 8 <-- the cost of each cell
```

Output

Print out the total cost to get from the starting cell to the destination cell. Example output for the given input above is shown below:

```
43
```

Skeleton Program

Your program is to be named **RobotNavigation.java** (do not change this name). A skeleton program is provided, but you can change it any way or add new class(es) and method(s) as you deem fit. **Write all your code in RobotNavigation.java, do not create new files!** The skeleton program is as follows:

```
import java.util.*;

/* class representing the Robot Navigation system */
public class RobotNavigation {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

    }
}
```

Testing your program

The following **sample** input and output files are in your plab account:

```
RobotNavigationTC1.in, RobotNavigationTC2.in, RobotNavigationTC3.in
RobotNavigationTC1.out, RobotNavigationTC2.out, RobotNavigationTC3.out
```

RobotNavigationTC1.in is the input test case for **part 1**, RobotNavigationTC2.in is the input test case for **part 2** and RobotNavigationTC3.in is the input test case for **part 3**. RobotNavigationTC1.out, RobotNavigationTC2.out, and RobotNavigationTC3.out are the expected output for the respective test cases.
After you have compiled your program, to test it with say RobotNavigationTC1.in, you type:

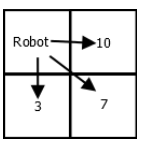
```
java RobotNavigation < RobotNavigationTC1.in
```

Please complete part 1 before starting on part 2 !

Problem Description - Part 2 (4 test cases/28 marks)

For this part, the hall is partitioned into a two dimensional grid and it is represented as an NxN square matrix. **The robot can start from any cell (starting cell given in the input) and it must move towards the bottom right cell which is the destination cell**. For this part, the robot can move in 3 possible directions

1. East - Move into cell right of current cell
2. South - Move into cell below current cell
3. South-East - Move into cell below and right (diagonal) of current cell



The navigation strategy of the robot is as follows

1. If none of the possible cells the robot can move to is the destination, then **greedily choose the cell with the lowest cost to move to**.
2. If one of the possible cells the robot can move to is the destination, then move to the destination cell.

In the example above, the robot will move down into the cell with cost 3. An example of how the robot moves in a 4x4 matrix, starting from cell (0,0) and going towards cell (3,3) is shown below (the bold and underlined cells are the path taken). The total cost is 10+0+8+30+9 = 57.

4x4 matrix

10	11	12	55
3	0	8	40
4	35	30	50
5	6	7	9

—> path of robot from (0,0) to (3,3)

10	11	12	55
3	**0**	**8**	40
4	35	**30**	50
5	6	7	**9**

Write a recursive method to compute the total cost of moving the robot from a starting cell to the destination cell (bottom right cell), given the above navigation strategy. You can assume that the total cost of moving the robot from the starting cell to the destination cell **will not exceed 1000000**, and that the **value in each cell is unique**.

Input

The 1st line in the input is the value 2. This indicates the input is for part 2. The next line is an integer **N** (**N** $\geq$ 1) indicating the dimension of the NxN matrix. The following line is the starting cell of the robot represented by row index followed by column index separated by a white space. The next N lines represent the N rows of the matrix. There are N integers in each line separated by a white space, and they represent the cost of each cell for that row. An example input is given below:

```
2   <-- indicates this is input for part 2
4   <-- value of N for the NxN matrix
0 0 <-- starts at row 0, column 0
10 11 12 55
3 0 8 40
4 35 30 50
5 6 7 9
```

Output

Same as before. Example output for the given input above is shown below:

```
57
```

Please complete part 2 before starting on part 3 !

Problem Description - Part 3 (2 test cases/14 marks)

For this part, it is basically the same as part 2. The robot can only move in the 3 possible directions as stated in part 2. However, instead of using the navigation strategy in part 2, the robot **will choose the minimum cost path to move from the starting cell to the destination cell**. Using the same example in part 2, the minimum cost path the robot will take to move from (0,0) to (3,3) is as shown. The total cost of this path is 10+3+4+6+7+9 = 39.

10	11	12	55
3	0	8	40
4	35	30	50
5	**6**	**7**	**9**

Write a recursive method to find the total cost of the minimum cost path to get from starting cell to the destination cell. You can assume that the total cost of the minimum cost path **will not exceed 1000000**, and that the **value in each cell is unique**.

Input

The 1st line in the input is the value 3. This indicates the input is for part 3. The rest is the same as part 2. An example input is given below:

```
3   <-- indicates this is input for part 3
4   <-- value of N for the NxN matrix
0 0 <-- starts at row 0, column 0
10 11 12 55
3 0 8 40
4 35 30 50
5 6 7 9
```

Output

Same as before. Example output for the given input above is shown below:

```
39
```

Grading Scheme

1. Program correctness = 70 marks, Design = 20 marks, Programming = 10 marks
2. No marks awarded if the program does not compile.
3. Marks will be deducted if **student particulars and program description** are not filled up in the top portion of the source code.
4. There are 10 test cases. Each test case is worth 7 marks.
5. Correctness marks will be not be given or will be halved as indicated in the requirements.
6. Things to look out for under design
   (a) Correct usage of programming constructs (eg use correct type for variables)
   (b) Not overly complicated logic
   (c) No redundant logic
7. Things to look out for under programming style
   (a) Meaningful comments
   (b) Pre-condition (if appropriate) should be present in the comment above a recursive method
   (c) Proper indentation
   (d) Meaningful identifiers