UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
School of Computer Science & Applied Mathematics

Introduction to Algorithms & Programming
Assignment 2

Due Date: 2 June 2023 23h55

1 Introduction

Like the labs, assignments are marked automatically online. This means that your program must produce *exactly* the expected output for it to be considered correct. For each task, submit the appropriate `.cpp` file to the Moodle marker. Successfully completing all five tasks will earn you 100%.

Again, please be aware of the policy surrounding plagiarism (see the course outline for a full description). Code will be scrutinised, and anyone found to have cheated (including the person from whom the code was copied) will immediately receive 0, and may be subject to disciplinary sanctions.

2 Background

For the past 12 years, we've run a competition for the second years in which they compete in games of multiplayer *Snake*. Remember *Snake*? No? The Nokia 3310? Nope?! Boy, am I old!

Anyway, for those who have never played *Snake* before, it's quite famous. It even has its own Wikipedia page (`https://en.wikipedia.org/wiki/Snake_(video_game)`). The essence of the game is that you're a snake moving around on a grid, and your goal is to eat as many apples as possible. Eating apples causes you to grow in length, so the game gets harder over time as you fill up more and more of the grid.

3 The Plan

The series of tasks that follow will get you pretty close to being able to compete in this year's competition. For those interested, the rules for last year's competition can be found here: `https://snake.wits.ai/docs`. At the end of this assignment, you should have a simplified version of the game, which contains the foundation you would need to compete in the full version.

4 The First Task (25 marks)

The first thing we should be able to do is represent and visualise the game's grid. Write a program that accepts the width and height of the grid, and prints the grid to the screen.

NB: Do not, at any point during the assignment, use variable-length arrays! It may run fine on your computer, but it could cause the marker to return a compilation error. Use dynamically-allocated arrays or (preferably) vectors instead.

4.1 Input

Input consists of a single line containing two integers — the width and height of the grid.

4.2 Output

Output a matrix where every element is zero. The number of columns should equal the width of the grid, and the number of rows the height. Ensure there's a single space between each element.

4.3 Example Input-Output Pairs

Sample Input #1

```
3 3
```

Sample Output #1

```
0 0 0
0 0 0
0 0 0
```

Sample Input #2

```
4 3
```

Sample Output #2

```
0 0 0 0
0 0 0 0
0 0 0 0
```

Sample Input #3

```
3 4
```

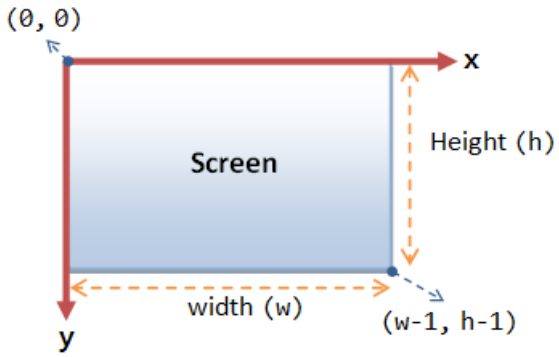
Sample Output #3

```
0 0 0
0 0 0
0 0 0
0 0 0
```

---

5 The Second Task (30 marks)

Now let's add a single snake and apple to the board. Write a program that accepts the width and height of the grid, as well as the location of an apple and snake and displays the grid on the screen. Note that the coordinates of the grid are illustrated by the following image:



The 2D Screen Coordinates: The origin is located at the top-left corner, with x-axis pointing left and y-axis pointing down.

Figure 1: Top left corner of the grid is represented by $x = 0, y = 0$, and the y-axis increases going down. Image from `https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html`

5.1 Input

The first line of input consists of two integers: the width and height of the grid. The second line of input consists of two integers: the x, y coordinates of the apple. The next line contains six integers, separated by a space. The first pair of integers specifies the x, y coordinates of the snake's head, the second pair the snake's body and the third its tail.

5.2 Output

Output the grid with the apple and snake drawn on it. The apple is represented by a 5, while the snake is represented by a 1. If there is no snake or apple at a particular location, a 0 should be used.

5.3 Example Input-Output Pairs

Sample Input #1

```
5 5
4 4
0 0 1 0 2 0
```

Sample Output #1

```
1 1 1 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 5
```

Sample Input #2

```
3 3
1 1
0 0 0 1 0 2
```

Sample Output #2

```
1 0 0
1 5 0
1 0 0
```

Sample Input #3

```
4 3
3 0
1 1 2 1 3 1
```

Sample Output #3

```
0 0 0 5
0 1 1 1
0 0 0 0
```

6 The Third Task (20 marks)

We now have the ability to represent a grid with an apple and snake positioned on it. However, we ultimately wish to play a multiplayer version of the game, so let's add a few more snakes. Write a program that accepts the width and height of the grid, the apple, and the location of multiple snakes and displays the resulting grid on screen.

6.1 Input

The first line of input consists of two integers: the width and height of the grid. The second line of input consists of two integers: the x, y coordinates of the apple. The next line contains a single integer N, representing the number of snakes that will be on the grid. You may assume that $N \leq 4$. N lines of input follow, each line representing a single snake and containing six integers, separated by a space. The first pair of integers on each line specifies the x, y coordinates of a snake's head, the second pair a snake's body and the third its tail.

6.2 Output

Output the grid with the apple and snake drawn on it. The apple is represented by a 5, while each snake is represented by an integer. The first snake is 1, the second snake is 2, and so on. If there is no snake or apple at a particular location, a 0 should be used.

6.3 Example Input-Output Pairs

Sample Input #1

```
5 5
4 4
3
0 0 1 0 2 0
4 0 4 1 4 2
0 2 1 2 2 2
```

Sample Output #1

```
1 1 1 0 2
0 0 0 0 2
3 3 3 0 2
0 0 0 0 0
0 0 0 0 5
```

Sample Input #2

```
3 3
1 1
1
0 0 0 1 0 2
```

Sample Output #2

```
1 0 0
1 5 0
1 0 0
```

Sample Input #3

```
4 3
3 0
2
1 2 2 2 3 2
1 1 2 1 3 1
```

Sample Output #3

```
0 0 0 5
0 2 2 2
0 1 1 1
```

7 The Fourth Task (15 marks)

We have now successfully represented the grid containing all the snakes and apples. Note that snakes can be identified by integers in the range $[1, 4]$, while the apple is 5, and empty space is 0.

The entire point of this is to create an agent that will actually play the game. So write a program that accepts the board description (as in the previous task) and outputs the best move available the first snake.

The snake has four available moves: `UP`, `DOWN`, `LEFT` and `RIGHT`. In order to calculate the best move, find the one that gets the snake to a valid grid cell that is closest (using straight-line distance) to the apple. Note that move selected would place the snake's head at that position, so ensure that the move does not result in a collision with any snake, or go off-grid.

7.1 Input

Input is in exactly the same format as The Third Task.

7.2 Output

Output the best move available to the first snake. If there are multiple moves that are equally as good, output all of them, ordered alphabetically, with one move per line. You may assume that there is always at least one move available.

7.3 Example Input-Output Pairs

Sample Input #1

```
5 5
4 4
3
0 0 1 0 2 0
4 0 4 1 4 2
0 2 1 2 2 2
```

Sample Output #1

```
DOWN
```

Sample Input #2

```
3 3
1 1
1
0 0 0 1 0 2
```

Sample Output #2

```
RIGHT
```

Sample Input #3

```
4 3
3 0
2
1 2 2 2 3 2
1 1 2 1 3 1
```

Sample Output #3

```
LEFT
```

8 The Fifth Task (10 marks)

Snakes grow when they eat apples, and so they will usually be quite long. Furthermore, they won't usually be in a straight line. All of this means that we need a different way to represent the snakes. Write a program that accepts the width and height of the grid, the apple, and the location of multiple snakes and displays the resulting grid on screen. Here, however, the input format will differ from that of The Third Task.

8.1 Input

The first line of input consists of two integers: the width and height of the grid. The second line of input consists of two integers: the x, y coordinates of the apple. The next line contains a single integer N, representing the number of snakes that will be on the grid. You may assume that $N \leq 4$. N lines of input follow, each line representing a single snake. Each line represents a snake's *coordinate chain* and takes the following format:

`k` x_1 y_1 x_2 y_2 `...` x_k y_k

The first integer is $k \geq 2$. k pairs of integers follow, each being a coordinate that represents a point of interest in the snake's body. The first coordinate is the snake's head, the next represents the first kink in the snake. There can be any number of kinks in the snake, all of which are all listed in order. Finally, the last coordinate represents the tail of the snake. For example, the line

```
4 0 0 2 0 2 5 6 5
```

represents a snake whose head is at $(0, 0)$, and whose body occupies all cells from $(0, 0)$ to $(2, 0)$, from $(2, 0)$ to $(2, 5)$, and from $(2, 5)$ to $(6, 5)$.

8.2 Output

Output the grid with the apple and snake drawn on it. The apple is represented by a 5, while each snake is represented by an integer. The first snake is 1, the second snake is 2, and so on. If there is no snake or apple at a particular location, a 0 should be used.

8.3 Example Input-Output Pairs

Sample Input #1

```
15 15
0 14
3
6 1 1 2 1 2 12 8 12 8 0 3 0
3 10 2 13 2 13 5
3 14 14 14 10 12 10
```

Sample Output #1

```
0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 1 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0 2 2 2 2 0
0 0 1 0 0 0 0 0 1 0 0 0 0 2 0
0 0 1 0 0 0 0 0 1 0 0 0 0 2 0
0 0 1 0 0 0 0 0 1 0 0 0 0 2 0
0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 0 3 3 3
0 0 1 0 0 0 0 0 1 0 0 0 0 0 3
0 0 1 1 1 1 1 1 1 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 3
5 0 0 0 0 0 0 0 0 0 0 0 0 0 3
```

9 The Bonus Task (∞ marks)

In the second semester, modify your code to adhere to this year's rules. Then participate in the *Snake* competition, and beat as many second years as possible.

9.1 Input

Your awesome program.

9.2 Output

Sad second years.