ASSIGNMENT 1

Subject : Array, Matrices
TAs: Merve Ozdes
Programming Lanuage: C++
Due Date: 5.11.2021 23:59

1 Introduction

In this assignment, you are expected to gain knowledge on C++ language including array and matrices.

2 Background

2.1 Arrays

C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element. To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows:

```
type arrayName [ arraySize ];
```

This is called a single-dimension array. The arraySize must be an integer constant greater than zero and type can be any valid C++ data type.

2.2 Multidimensional Array

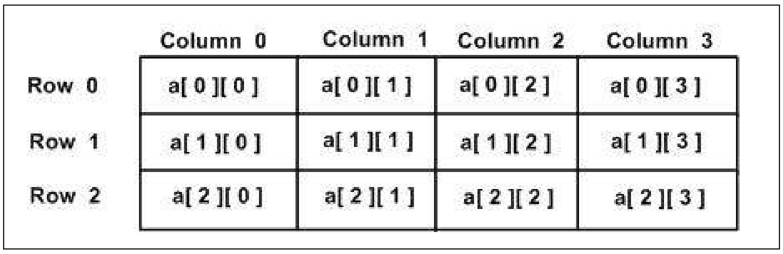
C++ allows multidimensional arrays. Here is the general form of a multidimensional array declaration

```
type name[size1][size2]...[sizeN];
```

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x,y, you would write something as follows:

```
type arrayName [ x ][ y ];
```

Where **type** can be any valid C++ data type and **arrayName** will be a valid C++ identifier. A two-dimensional array can be think as a table, which will have x number of rows and y number of columns. A 2-dimensional array a, which contains three rows and four columns can be shown as below



	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Figure 1: Two-dimensional array

3 Problem Definition

In this assignment, you are expected to simulate crazy balloon game. The grid consist of NxN cells (maximum value of N is 100), in the first part you will place balloons according to rules given and in second part, you will earn points by popping the balloons in a NxN grid given to you. There are several rules for this assignment, that are given below.

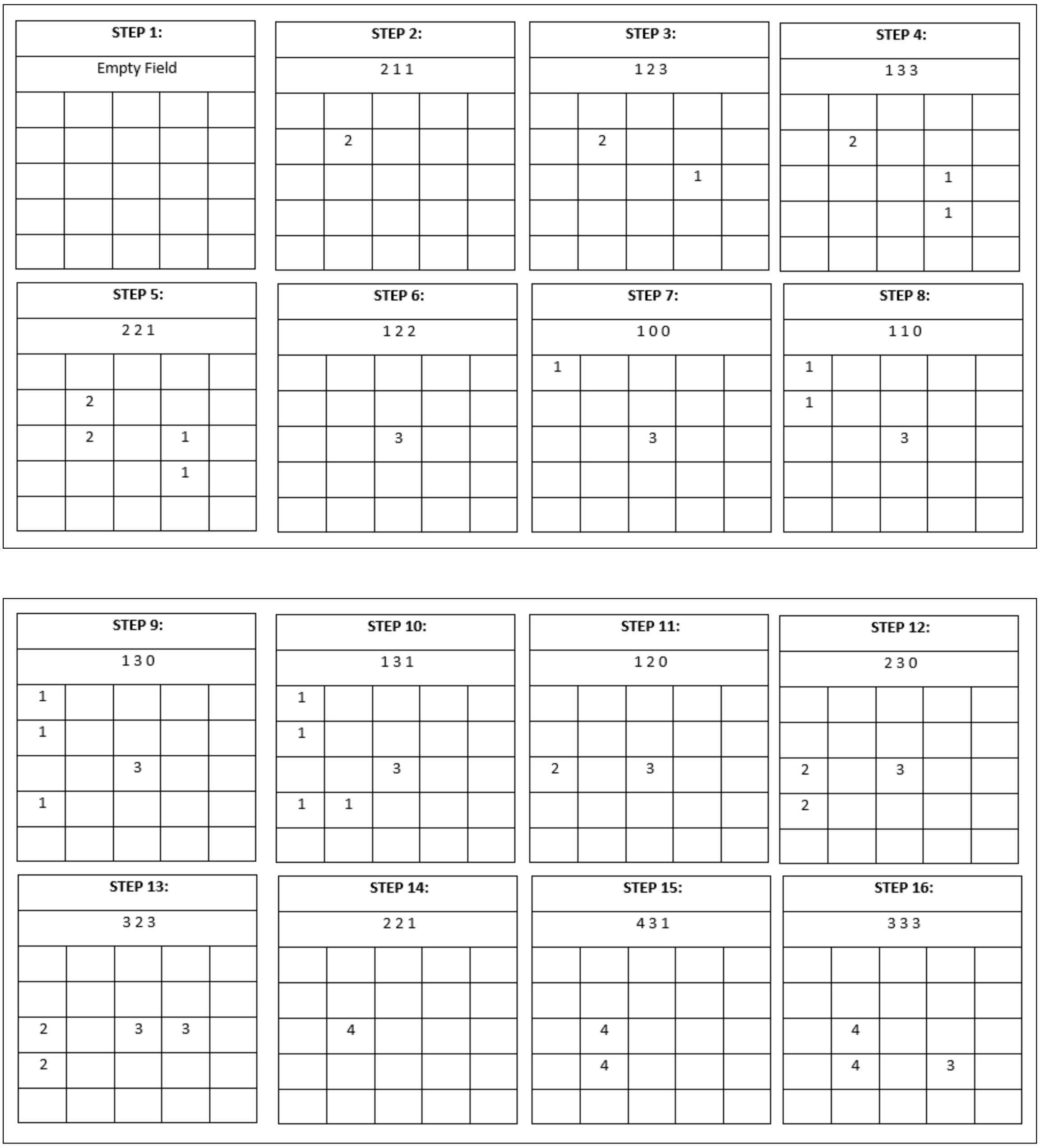
4 PART 1 - Placing of the Balloons

The grid will initially be given as empty. We will give type of balloons that place in each cell, respectively. We ask you to fill in the grid according to the rules we give below.

1. There are different types of balloons denoted by their type ID such as 1, 2, etc. For the next section, you should calculate the points of these balloons according to their type. Each type corresponds to the point of the balloons. If a balloon's type is t, it gives t points.
2. When a balloon of type x is place in a cell (current cell), if a link is created between **three or more balloons** of the same type, they will all be brought together in the current cell and a new balloon of type x+1 will be formed. The balloons making the link will disappear and these cells will have a value of 0.
3. A link can only be created between neighboring balloons. A cell's neighborhood consists of cells to its north, east, south, and west **(you don't need to think about diagonal relationships)**.
4. If creating new balloons of a different types triggers another link, you should also handle such cases.

4.1 Example: Construction of the Grid

The example of the construction of grid according to the input is shown below step by step. Example grid is given as 5x5 and the value of zero is not shown in cells.



STEP 1:

Empty Field

STEP 2:

2 1 1

	2			

STEP 3:

1 2 3

	2			
			1	

STEP 4:

1 3 3

	2			
			1	
			1	

STEP 5:

2 2 1

	2			
	2		1	
			1	

STEP 6:

1 2 2

		3		

STEP 7:

1 0 0

1				
		3		

STEP 8:

1 1 0

1				
1				
		3		

STEP 9:

1 3 0

1				
1				
		3		
1				

STEP 10:

1 3 1

1				
1				
		3		
1	1			

STEP 11:

1 2 0

2		3		

STEP 12:

2 3 0

2		3		
2				

STEP 13:

3 2 3

2		3	3	
2				

STEP 14:

2 2 1

	4			

STEP 15:

4 3 1

	4			
	4			

STEP 16:

3 3 3

	4			
	4		3	

Figure 2: Construction of the grid step by step

The explanation of each step that are given in Figure fig: step2 is given below:

- **Step 1:** Each cell has value 0 and 0 values are not shown in the figure.
- **Step 2:** 2 1 1 means put a balloon with type of 2 into the cell with 1st row, 1st column. Nothing happens because there is no link.
- **Step 3:** 1 2 3 means put a balloon with type of 1 into the cell with 2nd row, 3th column. Nothing happens because there is no link.
- **Step 4:** 1 3 3 means put a balloon with type of 1 into the cell with 3th row, 3th column. Nothing happens because there is no link.

- **Step 5:** 2 2 1 means put a balloon with type of 2 into the cell with 2nd row, 1st column. Nothing happens because there is no link.
- **Step 6:** 1 2 2 means to put a balloon type of 1 in the 2nd row, 2nd column. When we do this, a link emerges between three balloons of type x (x=1) and they will form balloon type of 2 (x+1=2) in cell [2,2]. Since having a with type of 2 in this cell also means that three balloons with type of 2 form a link, it will create a balloon with type of 3 in this cell [2,2]. This newly created balloon will be created in the current cell. The balloons that formed the link disappeared and their value became 0. Intermediate steps after linking are not shown. Only the balloon formed as a result of the link is written into the cell.
- **Step 7:** 1 0 0 means put a balloon with type of 1 into the cell with 0th row, 0th column. Nothing happens because there is no link.
- **Step 8:** 1 1 0 means put a balloon with type of 1 into the cell with 1st row, 0th column. Nothing happens because there is no link.
- **Step 9:** 1 3 0 means put a balloon with type of 1 into the cell with 3th row, 0th column. Nothing happens because there is no link.
- **Step 10:** 1 3 1 means put a balloon with type of 1 into the cell with 3th row, 1st column. Although we have four balloons with type of 1 , they are not linked after placing the existing one. Among them, the length of longest link is 2. So nothing happens.
- **Step 11:** 1 2 0 means put a balloon with type of 1 into the cell with 2nd row, 0th column. When we put balloon with type of 1 in cell [2,0], a five-length link appears between balloons with type of 1. As stated in step 6, the balloons that formed the link disappeared and newly created balloon (type of 2) placed in current cell[2,0].
- **........ (These cases are similar to previous steps.)**
- **Step 14:** When we put a balloon with type of 2 into the cell [2,1], then a link appears between balloons with type of 2 and a new balloon with type of 3 is formed. This newly created balloon triggers a new link between balloon with type of 3 and balloon with type of 4 is formed in the current cell [2,1].
- .....
- **Step 16:** There is no link between balloons.

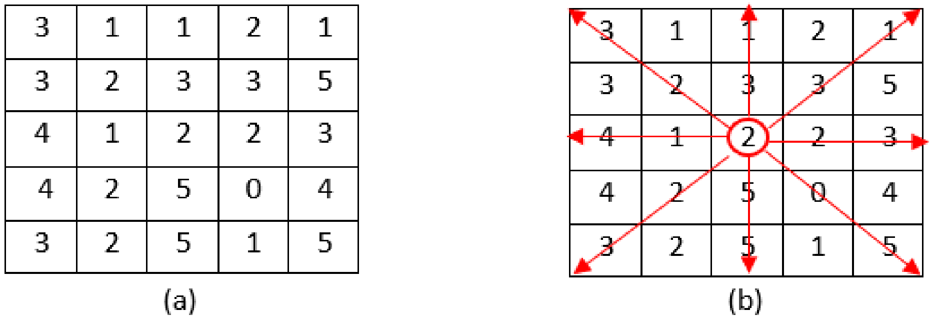
5 Part 2: Putting the Bomb

In the second part, it is expected that you will pop the balloons with the help of a bomb placed in the grid given to you and calculate the points of the popped balloons. Where to place the bombs will be given in the input2.txt file. Popping the balloons with bombs can be with balloons on the north, east, south, west and diagonal line of the bomb cell. After the bomb is placed in the grid, the balloons in the north, east, south, west and diagonal lines of the bomb and the same value as the bomb's id are popped. 0 is written in the cells of the popped balloons. Bombs can also be placed in cells with popped balloons, that is, cells with a value of 0 and also bombs can be placed in the same cell more than once.

5.1 Example: Putting the Bomb

The example of putting the bomb and popping the balloons according to the cell of the bomb is given below. Rules of putting the bomb and calculating final point:

- When you put the bomb in any cell, the value of the balloon in the cell where the bomb is placed and the cells of the same value on the north, east, south, west and diagonal lines will be replaced with 0. For example, you placed a bomb in cell [2,2] as shown in Figure 3(b).



(a)

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

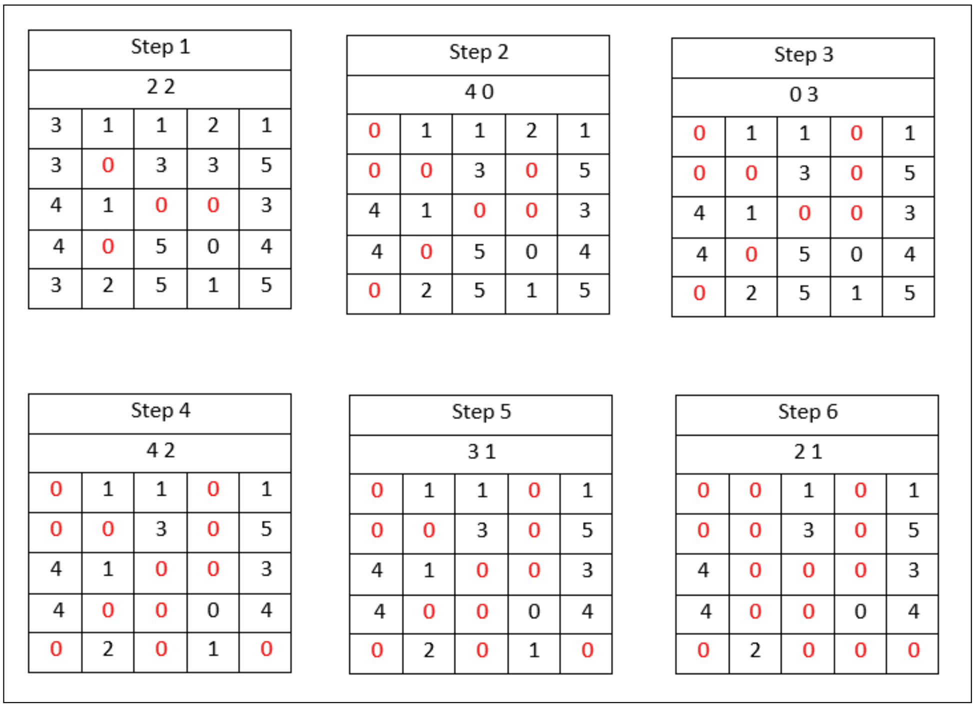
(b)

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

Figure 3: a) NxN grid b) East, West, North, South and Diagonal lines of the bomb

You have to pop all the balloons that are equal to the value of the balloon in the cell where you placed the bomb and are in the north, east, south, west and diagonal lines. As shown in Figure 3(b), balloons in cells [1,1], [2,2], [2,3] and [3,1] will be popped.

- After the bomb is placed, the value of all the balloons that is popped is added up and the point is calculated. These points are added up each time a bomb is placed to obtain the final point.
- After the bomb is placed, if there is no match between the value of bomb cell and balloons on north, east, south, west and diagonal lines, only the balloon in the cell where the balloon is placed pops (e.g. Step 3).
- A bomb can also be placed in the cell where the previously popped balloon was located. In such a case, there will be no change (e.g. Step 5).



Step 1				
2 2				
3	1	1	2	1
3	0	3	3	5
4	1	0	0	3
4	0	5	0	4
3	2	5	1	5

Step 2				
4 0				
0	1	1	2	1
0	0	3	0	5
4	1	0	0	3
4	0	5	0	4
0	2	5	1	5

Step 3				
0 3				
0	1	1	0	1
0	0	3	0	5
4	1	0	0	3
4	0	5	0	4
0	2	5	1	5

Step 4				
4 2				
0	1	1	0	1
0	0	3	0	5
4	1	0	0	3
4	0	0	0	4
0	2	0	1	0

Step 5				
3 1				
0	1	1	0	1
0	0	3	0	5
4	1	0	0	3
4	0	0	0	4
0	2	0	1	0

Step 6				
2 1				
0	0	1	0	1
0	0	3	0	5
4	0	0	0	3
4	0	0	0	4
0	2	0	0	0

Figure 4: Putting the bomb step by step

- Each step point is as shown below:
  - Step 1: 8p (4*2)
  - Step 2: 12p (4*3)
  - Step 3: 2p (1*2)
  - Step 4: 15p (3*5)
  - Step 5: 0p (There is no popped balloon)
  - Step 6: 3p (3*1)
  - Final Point: 40p

6 Inputs and Outputs

For this assignment you have two input file. You are expected to produce output.txt file according to input1.txt file and input2.txt file. The format of input and output files are shown below. In the input1.txt, the first line is the N value indicating the size of the grid, and the maximum value of N is 100. The next lines show the incoming balloons in <t> <x> <y> format. Here, t is type of the incoming balloon, x and y are the indices of the cell it will place. The input1.txt file of the example given in Figure 2 is as follows.

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

In the input2.txt file, you are given NxN grid and you will place the bomb to specified cell. First line which is N indicates size of the NxN grid. And there will be N line, each line has N integers separated by a space character. Following N lines, each line gives the indices of the cell in which bomb will be placed. Each row is in <x> <y> format. Here x and y are the indices of the cell bomb will place. An input of example grid shown in Figure 3 and Figure 4 are as follows:

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

The output of this assignment which are created according to the input1.txt file and input2.txt file should be as shown below. When printing the NxN grid of the last step, there should be a space between each column and a blank line between part1 and part2. Please create your output file according to the format given to you.

```
PART 1:
0 0 0 0 0
0 0 0 0 0
0 4 0 0 0
0 4 0 3 0
0 0 0 0 0

PART 2:
0 0 1 0 1
0 0 3 0 5
4 0 0 0 3
4 0 0 0 4
0 2 0 0 0
Final Point: 40p
```

7 Grading and Evaluation

- Your work will be graded over a maximum of 100 points.
- Your total score will be partial according to the grading policy stated below.

Part 1	50p
Part 2	40p
Code design, clean and readable code, algorithmic perspective, comments	10p

- **Your code will be tested on dev platform with different inputs. And one output file will be expected as output file.**
  - Upload your files to your server account (dev.cs.hacettepe.edu.tr)
  - Compile your code (g++ -o Main *.cpp)
  - Run your program (./Main input1.txt input2.txt output.txt)
  - Control your output (output.txt).

Usage example:

```
>g++ -o Main *.cpp
>./Main input1.txt input2.txt output.txt
```

Notes

- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Write READABLE SOURCE CODE block
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/fall/bbm203) and you are supposed to be aware of everything discussed in Piazza.
- You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ No other submission method (email or etc.) will be accepted. Do not submit any file via e-mail related with this assignment.
- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system). You must submit your work with the file hierarchy stated below:

```
<student_id.zip>/(Required)
    →src/(Required)
        →Main.cpp(Required)
        →*.cpp(optional)
        →*.h(optional)
```

Policy

All work on assignments must be done **individually** unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an **abstract** way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) **will not be tolerated**. In short, turning in someone else's work (from internet), in whole or in part, as your own will be considered **as a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.