COP 3014 Fall 2021
Homework 5

Total Points: 100
Due: Monday 11/01/2021

1 Objective

The objective for this assignment is to make sure

- You understand and can work with C++ arrays.
- You are comfortable with writing functions in C++.
- You are familiar with repetitive structures (loops) and selection statements (if/else or switch) in any combination and can use them in functions and to manipulate array data.
- You can approach a complex problem, break it down into various parts, and put together a solution.

For this assignment, please make sure you conform to Output requirements. You should now be familiar enough with cout statements and output formatting that you should be able to EXACTLY match the sample output (other than certain exceptions you will be informed about beforehand).

This assignment requires you to submit multiple files on Canvas. Please do so in a SINGLE submission. Canvas allows you to turn in multiple flies in one submission. Once you have uploaded your first file, click "attach another file" to upload your second file, and so on.

Turn in your files `arrays.cpp` and `matrix.cpp`, through Canvas.

2 A Note on I/O in functions, and Testing

2.1 Input and Output in a Program with Functions

Ideally, functions are written to perform an operation, and thee main function would handle all interactions with the user. Unless it is specifically a function written for input or output, a function should not have any `cout` or `cin` statements. We will hold to this convention in these programs.

- The input function (whatever you choose to call it) and the main function should be the only functions with `cin` statements.
- The output function (whatever you choose to call it) and the main function should be the only functions with `cout` statements.
- If it a part of the operations of the function, then it should be in the function - main should only be used for user interaction (a menu if required) and to call the functions.
- You may write additional helper functions if required.
- Not adhering to these conventions would result in a loss of 5 points.

2.2 Testing

The sample runs we give you here are only for indicting the general functionality of the program. The student is responsible for testing their programs thoroughly for a variety of inputs and seeing if the output makes sense in context. Just matching the sample runs is not enough.

When you attend Help Sessions, the instructor and the TA'/LA's are not running your program. We're eyeballing it and will let you know if it's sound in concept, and if there is something catastrophically wrong. A TA saying "it's OK" during Help Sessions means exactly that - it *looks* alright, but we would have to run it and grade it to see what you're grade would be. You cannot ask for your work to be graded before the deadline.

3 Program 1 - Array Operations - 60 Points

Squilliam Fancyson has heard of the fancy calculator that Squidward (supposedly) made for the Krusty Krab's Cash Register. Determined to prove that he is the best programmer in town, he has challenged Squidward to a series of programming challenges. If Squilliam wins, Squidward has to hand over his clarinet and never paint again. If Squidward wins, Squilliam will work Squidward's shift at the Krusty Krab's cash register for a week. Squidward is determined to win, except he is not really a programmer. He has left no stone (pioneer transport or otherwise) unturned. He has even tried asking Karen, who was unable to help. He realizes that instead of working off Mr. Krabs' old computer (the one Patrick smashed while painting the house), he could just hire you to do the challenges for him (Squilliam did not expressly forbid this). Your task today is to help Squidward retain his sanity and dignity (whatever is left of it).

The first challenge is array operations. For this program, we're going to perform a range of standard array operations to show Squilliam once and for all, that Squidward is familiar with arrays and how they work. You're required to write functions to insert and delete elements from an array, and shift elements in the array. We're also going to write a command line menu to do these tasks repeatedly. For this problem, you can assume that the array size will never drop below 1 and will never rise above 99.

Please make sure you conform to the following requirements:

1. Create a global constant integer called CAPACITY and set it to 100 (5 points)
2. Use the class examples and exercises to write *functions* to initialize the array by reading values from the user, and to print the array. These functions take the array and its current size as parameters. You can use the code in the examples. (7 points)
3. Write a function called `insertAfter` that takes the array, a number to be inserted, a number to insert before and the current size of the array as parameters. Insert the second parameter after every occurrence of the third parameter. If the third parameter doesn't exist in the array, do not change the array. (13 points)
4. Write a function called `removeLast` that takes the array, a number and the current size of the array, and removes the last the instances of the number from the array. If the number does not exist in the array, do not change the array. (8 points)
5. Write a function called `cyclicRightShift` that takes the array, the shift quantity 'q' and its size as parameters and shifts all elements 'q' positions to the right. It should also move the last 'q' elements to the beginning of the array instead of removing them. You may assume that 'q' will always be smaller than the current size of the array. (9 points).

6. In the main function, create an array using the CAPACITY constant for capacity. (3 points)
7. Have the user initialize the array. Then, write a command line menu with the following options:
   - Insert
   - Remove
   - Print
   - Cyclic Right Shift
   - Exit

   Call the appropriate functions when the user chooses a particular option. Print an error message and continue if the user enters a wrong option. (10 points)
8. You can assume that the array will always contain at least 1 element and at most 99 elements.
9. Please make sure your code is appropriately commented. (5 points)

3.1 Sample Runs

```
Enter the number of elements you want to enter (Max 100): 7
Enter 7 numbers
2.5     0.8     6       -7      2.5      10.7    0.8

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 1
Enter the number: 3.1
Enter the number to be inserted after: 0.8
Element Inserted.

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 3

The array is:
2.5  0.8  3.1  6  -7  2.5  10.7  0.8  3.1

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 2
Enter the element to be removed: 2.5
```

```
Element deleted.

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 3

The array is:
2.5  0.8  3.1  6  -7  10.7  0.8  3.1

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 4
Enter the shift quantity 2
Elements shifted.

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 3

The array is:
0.8  3.1  2.5  0.8  3.1  6  -7  10.7

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 9
Invalid Choice.

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right Shift
5. Exit
Enter your option: 1
Enter the number: 4.5
Enter the number to be inserted after: 9.8

1. Insert an element
```

```
2. Remove an element
3. Print the array
4. Cyclic Right Shift
5. Exit
Enter your option: 2
Enter the element to be removed: 8.25

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right Shift
5. Exit
Enter your option: 3
The array is:
0.8  3.1  2.5  0.8  3.1  6  -7  10.7

1. Insert an element
2. Remove an element
3. Print the array
4. Cyclic Right shift
5. Exit
Enter your option: 5
Goodbye!
```

4 Program 2 - Weighted Matrix Average - 40 points

The competition is heating up. The special judging panel, consisting of Karen, Old Man Jenkins and Sandy, do not want to release the results of the previous challenge until this one is complete as well. For this challenge, you have to compute the weighted average value of a matrix. This value is defined as

$$WeightedAverage = \frac{1}{m * n} \sum_{j=]}^{n} \sum_{i=1}^{m} A_{ij} * W_j$$

. Where W_j is a weight assigned to each column. Write a C++ program to compute the weighted average.

Please make sure you conform to the following requirements:

1. Declare global constant integers for ROWCAP and COLCAP. Set both to 100 (3 points)
2. Use the class examples and exercises to write *functions* to initialize the matrix by reading values from the user, and to print the matrix. These functions take the matrix and its current number of rows and columns as parameters. You can use the code in the examples. (4 points)
3. Write a function called `findAverage` that accepts the matrix, its current number of rows and columns and returns its weighted average. (7 points)
4. In the main function, create the matrix, read in the number of rows and columns and call the function to read in the matrix values. (4 points)
5. Read in the weights of each column and calculate the weighted matrix in-place. That is, do not create a new matrix. Modify the original matrix to replace the initial values with the weighted values. At the same time, calculate the total weight by keeping a running sum. (9 points).

6. Call the print function to print the weighted matrix. (3 points)

7. compute and print the weighted average by calling the `findAverage` function. Also print the average weight. You do not have to account for floating point precision. (6 points)

8. Please make sure your code is commented (4 points).

4.1 Sample Run

```
Enter the number of rows: 4
Enter the number of columns: 5
Enter the matrix:
-3   5   17    16   25
2    6.2    4    9   -18
0    2    8    51   4.25
-9   98   -14  41   3
Enter the column weights: 3  1.5  2  4  2.7
The weighted matrix is:
-9      7.5     34      64      67.5
6       9.3     8       36      -48.6
0       3       16      204     11.475
-27     147     -28     164     8.1

The average weight is 2.64
The average weighted value is 33.6638
```

5 Generic Guidelines

1. Include the header comment with your name and other information on the top of your files.

2. Please make sure you're only using the concepts already discussed in class. Please restrict yourself to input/output statements, variables and operators, selection statements, loops, functions, and arrays. Using strings, pointer, structs, generic classes, iterators or anything more advanced will result in a loss of 25 points.

3. If we have listed a specification and allocated point for it, you will lose points if that particular item is missing from your code, even if it is trivial.

4. No global variables (variables outside of main())

5. No use of the `auto keyword` or any other C++ 11 or higher features.

6. Functions have to be declared above main and defined below main. Not doing so will result in a loss of 10 points.

7. You need to ensure you do not go out of bounds in the arrays, even if CLion protects your program from crashing.

8. You need to follow the other programming conventions established in the course. This includes not using break or continue statements, unless the break statement is used to avoid the fall-through in a switch statement.

9. **This is individual work. You may NOT collaborate with other students in the course, former students, hire tutors to "help", copy solutions off the internet, or use pay-for solution websites, including but not limited to Chegg, CourseHero, WiseAnt, Bartelby, tutor.com, assorted Social Media groups, whatever GroupMe students might have created, etc.) This includes posting the problem statements on these websites even if you do not use the answer obtained. Doing so is a violation of the Academic Honor Code. Violation of the Honor Code will result in a 0 grade on the program, a reduced letter grade in the course and potentially more serious consequences.**
10. All input and output must be done with streams, using the library `iostream`
11. You may only use the `iostream` library (you do not need any others for these tasks)
12. Please make sure that you're conforming to specifications (program name, function names and parameters, print statements, expected inputs and outputs etc.). Not doing so will result in a loss of points
13. NO C style printing is permitted. (Aka, don't use printf). Use cout if you need to print to the screen.
14. When you write source code, it should be readable and well-documented (comments).
15. Make sure you either develop with or test with Jetbrains CLion (to be sure it reports no compile errors or warnings) before you submit the program.
16. Testing your program thoroughly is a part of writing good code. We give you sample runs to make sure you match our output requirements and to get a general idea of how we would test your code. Matching your outputs for JUST the sample runs is not a guarantee of a 100. We have several extensive test cases.
17. Please make sure you've compiled and run your program before you turn it in. Compilation errors can be quite costly. We take 5 points off per compiler error for the first 9 errors. The 10th compiler error will result in a grade of 0.
18. Only a file turned in through Canvas counts as a submission. A file on you computer, even if it hasn't been edited after the deadline, does not count.
19. The student is responsible for making sure they have turned in the right file(s). We will not accept any excuses about inadvertently modifying or deleting files, or turning in the wrong files.
20. **Program submissions** should be done through the Canvas class page, under the assignments tab (if it's not there yet I'll create it soon.) Do not send program submissions through e-mail – e-mail attachments will not be accepted as valid submissions.
21. The ONLY files you will submit via Canvas are `arrays.cpp` and `matrix.cpp`.
22. **General Advice** - always keep an untouched copy of your finished homework files in your email. These files will have a time-stamp which will show when they were last worked on and will serve as a backup in case you ever have legitimate problems with submitting files through Canvas. Do this for ALL programs.