Lab 6
Introduction to C++ and Classes.

Objectives

1. Become familiar with Classes and Objects
2. Work with Constructors
3. Learn about overloaded constructors and functions.
4. Exception handling.

Prelab Assignments

- If not done, finish the reading assignments posted on Blackboard.
- Make sure you have working code from lab 4.

The main idea of this lab is to convert your lab 4 code into a C++ code, defining classes with corresponding data members and methods to perform the desired signal operations.

Tasks:

Create a *Signal* class that contains the following / has the following properties:

1. Signal info. There should be data members for: **length** (i.e. # of data points), **maximum value** and **average**.
2. A pointer to the signal data.
3. The above data members should be private.
4. A *default* empty constructor, and two *parametric* constructors.
   - An empty constructor should take a default input data file and allocate the memory accordingly.
   - For one of the parametric constructors you should pass a file number. The base file name will be assumed to be as in lab 4: “Raw_data_nn.txt”. The constructor will read the input signal and allocate the required memory.
   - For the other parametric constructor, you should pass a file name. The file does not need to be named “Raw_data_nn.txt”, but it should have the same format. The constructor will read the input signal and allocate the required memory.

5. Exception Handling: All input values supplied by the user should be checked to see if they are valid. If not, ask for valid inputs and/or display error messages.
6. Create a destructor to de-allocate the memory.
7. Your functions (Offset, Scale, Center, Normalize, Statistics) from lab 4 should be converted into public methods of the class (member functions). This time, as opposed to lab 4, when those functions are called, no text file will be created. Instead, the signal itself and/or the other data members of the object will be updated. Note that some of the methods may require input parameters, and some may not. Note also that if an offset (or another operation) is performed, the signal info data members may need to be updated accordingly.

   You may need to create auxiliary methods (e.g. a function that calculates averages). These auxiliary functions should be private members.
8. The following additional methods should be implemented:
   - Sig_info: to display the length, the current maximum value and the current average of the signal.
   - Save_file: to create a text file with the current signal data. The method should take a single input string with the name of the file to be created. The format of the file should be the same as the "Raw_data_nn.txt", where the maximum value and the data points are the current ones.

Your program may take some command line arguments: **–n file_number** or **–f file_name**, to create a signal object. If there is no command line input, you may ask the user to enter a file name, or you may call the default constructor. Then, the program should go into a loop, asking the user what he/she wants to do next (show the user a menu, accept inputs). Based on the user input, the appropriate actions should be taken. Use **cout** and **cin** to display messages and read input, respectively.

Push your code to your github repository.

Experiments:

Run various cases, using various files. Create files using the Save_file method, then use those as input files. Make sure to test all your methods.

Deliverables:

1. Lab Report:
    1) The report should follow the lab report format posted on Blackboard.
    2) In the results section, show a few test cases and include screenshots of the program handling inputs and displaying results.
    3) Discussion: Make sure to discuss your results and to describe any problems and interesting things that you may have encountered while coding the lab, especially while converting your previous code into classes and methods.
    4) Screenshot of your github repository showing your commit.
    5) Append your source code with proper comments to the end of you report.

2. Demonstration:

    Show a few output cases to the TA. Perform some operations, display the signal info, and show saved files.

Grading:

Since this lab is mostly based on converting your previous code to classes, an important part of the grade will be given to your code.

Demo:	20	
Report:	40	
Code:	40	**(Well organized, properly commented and indented)**