Lab 2: Pointer, Reference, Dynamic Memory Allocation and Release, File I/O, and Makefile

Objectives:

- Using **function pointer**.
- Using **pointers**, **references**, and **dynamic memory allocation**.
- Using **ifstream** and **ofstrem** to read from and write into files.
- Organizing C++ application into appropriate folders.
- Separating the **specification** from the **implementation**.
- Writing the application's main function in a separate file.
- Generate separate object files from all the source files.
- Linking the object files into single executable file.
- Writing **Makefile** to aid in building the application.
- Building (compiling and linking) the application using **make**.

Tasks:

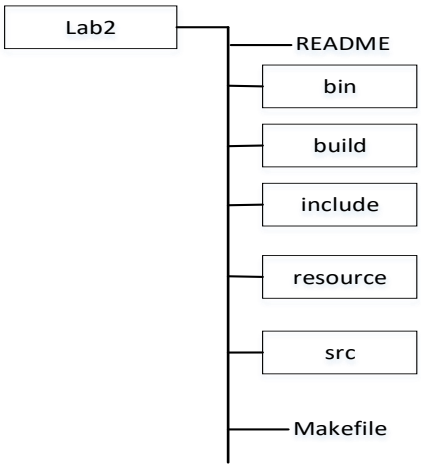
1. You will submit this lab using **GIT submission system**. A central repository named '**lab2**" has been created for this lab. Create your own fork of **lab2** on the central GIT repository using following command. Replace ***NN*** part of the command by the last two digits of your section number. For example, if your section number is ***S20N01***, replace ***NN*** by ***01***.

   *ssh csci fork csci161-**NN**/lab2 csci161-**NN**/$USER/lab2*

2. You have created a folder named **csci161-NN** in your home folder in lab1. Please, don't forget to replace the ***NN*** part of the folder name by the last two digits of your section number.
3. Go into your **csci161-NN** folder and create a clone of your forked **lab2** repository using following command. Again, replace ***NN*** part of the command by the last two digits of your section number.

   *git clone csci:csci161-**NN**/$USER/lab2*

4. Repository **lab2** has been organized as follows:



A **README** file template has been placed in the root of the application development folder. The README file gives a general idea of the application, technologies used in developing the application, how to build and install the application, how to use the application, list of contributors to the application, and what type of license is given to the users to use the application. You need to complete the README file.

All **header** files (**filedata.h** and **operation.h**) are placed in **include** sub folder.

You need to place your **source codes** (**filedata.cpp**, **operation.cpp**, and **main.cpp)** in **src** sub folder**.**

A sample data file named **data.txt** is placed in **resource** folder. It contains **integer numbers** in a single line. The **first number** in the line represents the number of integer data in the file, i.e., the size of the data and it is not included in the data.

All **object** files will be placed in **build** sub folder and **executable** files in **bin** sub folder by **make**. A place holder **Makefile** has also been supplied. You need to complete the **Makefile** to build your application in this lab.

1. Write **filedata.cpp** file in **src** folder to implement all the functions specified in **filedata.h** file.
2. Write **operation.cpp** file in the **src** folder to implement all the functions specified in **operation.h** file.
3. Write **main.cpp** file in **src** folder with a **main()** function. The **main()** function in **main.cpp** must do the followings:
   a) Must check whether the application is called with the command line argument specifying the paths for input and output file. If the file paths are missing, it should display usage message and exit.
   b) Call **load()** function from '**filedata.cpp**' to the data from the input file into an array and to get the size of the data.
   c) Show the size of the data on the screen.
   d) Show the array elements on the screen using **show()** function of '**operation.cpp**'.
   e) Add 10 to each element of the array by calling **operate()** function of '**operation.cpp**' and passing **add()** function as the function pointer into **operate(**) function.
   f) Show the array elements on the screen using **show()** function of '**operation.cpp**'.
   g) Subtract 10 from each element of the array by calling **operate()** function of '**operation.cpp**' and passing **subtract()** function as the function pointer into **operate(**) function.
   h) Show the array elements on the screen using **show()** function of '**operation.cpp**'.
   i) Multiply 10 with each element of the array by calling **operate()** function of '**operation.cpp**' and passing **multiply()** function as the function pointer into **operate(**) function.
   j) Show the array elements on the screen using **show()** function of '**operation.cpp**'.
   k) Save the multiplied array elements into the output file calling '**save()**' function of '**filedata.cpp**'.
   l) Release all dynamically allocated memory before return.

4. Write a **Makefile** in the **root** folder of the application which will do the followings:
   a) Define and use **macros** for each **GCC flag** that you are going to use to compile/link your code/object.
   b) Define and use **macros** for each **sub-folder** of the application, e.g., **src**, **include**, **resource**, **build**, and **bin**.
   c) Use GCC **debug flag** to facilitate debugging using **gdb**.
   d) Use GCC **include flag** to specify the path of application's custom header files so that your code does not need to specify relative path of these header files in **#include** pre-processor macro.
   e) Create individual object file (***.o**) into **build** folder from each ***.cpp** file of **src** folder.
   f) Link all the object files at the **build** folder into a single executable file named **lab2** into **bin** folder.

g) Clean or remove files from both **build** and **bin** folders using **PHONY target** named **clean**.

5. Continue your work in your cloned or local **lab2** repository and **commit** and **push** your work to your central **lab2** repository as it progresses.
6. Make sure your program compiles and runs error and warning free.
7. Test your program to make sure your code has fulfilled the specifications. Your program must work with any test data file.
8. You can run the example executable (**lab2**) from **bin** sub folder to get an idea what is expected from you in this lab. These example executable have been built and tested in Linux Debian machines available in the lab. Run these executable in other kind of machines at your own risks. Be careful to use **make clean**, it will delete the example executable from **bin** sub folder. You can save these example executable in another sub folder, for example, **backup,** in order to have a copy of them.
9. You should type following at the command prompt to run your executable:

   >bin/lab2 resource/data.txt resource/multiplied.txt

10. Organize and comment your code to make it easy to understand. Make sure you have typed **your name** and **student number** in the top comment section in each **.cpp** file. Make sure you have deleted all debug print codes that you were using to debug your code during your development time but not necessary in the final code. Make sure you have deleted all commented out codes from your final submission.

Deadline and Submission

The deadline to submit this lab is **4:00 PM** on **January 31, 2020 (Friday)** for **S20N01** and **S20N03** and **February 03, 2020 (Monday)** for **S20N02.**

Commit and push your work from your local lab2 repository to your remote lab2 repository regularly. You will find most useful git commands in this git cheat sheet from GitLab. You will be allowed to commit and push until the deadline is over. Incremental and frequent commits and pushes are highly expected and recommended in this lab.

Evaluation

Implementations	Features	Marks
	load() function	**10**
	save() function	**10**
	main() function	**10**
	show() function	**05**
	operate() function	**10**
	add() function	**05**
	subtract() function	**05**
	multiply() function	**05**
Makefile		**15**
README		**05**
Code Quality and Comments		**20**
Total		**100**