Assignment 4: Class, Object, Operator Overloading, Friend Function, Inheritance, Runtime Polymorphism, File I/O, and Makefile

Objectives:

- Creating and using **Dynamic Array class**.
- Creating and using **Simple Set class**.
- Creating **class constructors**, **destructors**, **copy and move constructors**, and **copy and move assignment operators**.
- **Overloading operators** (**+**, **-**, *****, **++**, **--**, **[]**, **>>**, **<<**).
- Using **friend function.**
- Using **inheritance.**
- **Overriding inherited** function**.**
- Using **runtime polymorphism** with **virtual function.**
- 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:

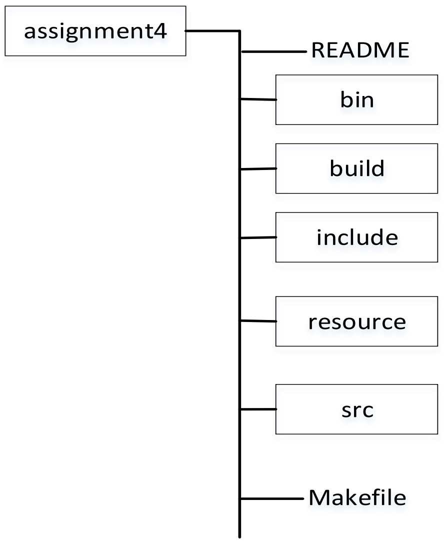
1. You will submit this lab using **GIT submission system**. A central repository named '**assignment4**" has been created for this lab. Create your own fork of **assignment4** on the central GIT repository using following command. Replace ***NN*** part of the command by the last two digits of your section number. For example, if your section number is ***S20N01***, replace ***NN*** by ***01***.

   *ssh csci fork csci161-*NN*/assignment4 csci161-*NN*/$USER/assignment4*

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 **assignment4** 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/assignment4*

4. Repository **assignment4** 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. <u>You need to complete the README file</u>.

All **header** files (**dynamicarray.h**, **simpleset.h**, **fileutility.h**, and **tests.h**) are placed in **include** sub folder.

You need to place your **source codes** (**dynamicarray.cpp**, **simpleset.cpp**, **fileutility.cpp**, **dynamicarraytests.cpp**, **simplesettests.cpp**, and **main.cpp)** in **src** sub folder**.**

Sample data files named **primes.txt** and **fibonacci.txt** have been placed in **resource** folder. Data file **primes.txt** contains first **10 numbers** in the **prime series** in a single line separated by spaces. Data file **fibonacci.txt** contains first **10 numbers** in the **Fibonacci series** in a single line separated by spaces.

All **object** files will be placed in **build** sub folder and the **executable** file in **bin** sub folder by **make**. A **Makefile** has also been supplied. <u>You use the supplied **Makefile** to build your application in this lab</u>.

1. Write **dynamicarray.cpp** file in the **src** folder to implement all the functions specified in **dynamicarray.h** file.
2. Write **simpleset.cpp** file in the **src** folder to implement all the functions specified in **simpleset.h** file.
3. Write **fileutility.cpp** file in the **src** folder to implement all the functions specified in **fileutility.h** file.
4. A **dynamicarraytests.cpp** file in **src** folder has been supplied, which has implemented the **test()** function of **tests.h** file. Use the supplied **dynamicarraytests.cpp**.
5. A **simplesettests.cpp** file in **src** folder has been supplied, which has implemented the **test()** function of **tests.h** file. Use the supplied **simplesettests.cpp**.
6. A **main.cpp** file in **src** folder with a **main()** function has been supplied. Use the supplied **main.cpp**.
7. Supplied **Makefile** in the **root** folder of the application does the followings:
   a) Defines and uses **macros** for each **GCC flag** that you are going to use to compile/link your code/object.
   b) Defines and uses **macros** for each **sub-folder** of the application, e.g., **src**, **include**, **resource**, **build**, and **bin**.
   c) Uses GCC **debug flag** to facilitate debugging using **gdb**.

d) Uses 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) Creates individual object file (***.o**) into **build** folder from each *.**cpp** file of **src** folder.
a) Links selected object files from the build folder into multiple **executable** files into **bin** folder. Each executable file in the bin folder either test dynamic array class or simple set class.
b) Cleans or removes files from both **build** and **bin** folders using **PHONY target** named **clean**.

5. Continue your work in your cloned or local **assignment4** repository and **commit** and **push** your work to your central **assignment4** 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 executables (**dynamicarray** and **simpleset**) from **bin** sub folder to get an idea what is expected from you in this assignment. These example executables have been built and tested in Linux Debian machines available in the lab. Run these executables in other kind of machines at your own risks. Be careful to use **make clean**, it will delete the example executables from **bin** sub folder. You can save these example executables 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/dynamicarray resource/fibonacci.txt resource/arrayoutput.txt

>bin/simpleset resource/primes.txt resource/fibonacci.txt resource/setoutput.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 **March 23, 2020 (Friday)** for **S20N01** and **S20N03** and **March 25, 2020 (Monday)** for **S20N02.**

Commit and push your work from your local assignment4 repository to your remote assignment4 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**
	DynamicArray.default-constructor	**01**
	DynamicArray.regular-constructor	**01**
	DynamicArray.copy-constructor	**02**
	DynamicArray.move-constructor	**02**
	DynamicArray.copy-assignment	**02**
	DynamicArray.move-assignment	**02**
	DynamicArray.destructor	**02**
	DynamicArray.empty	**01**

Implementations	**Features**	**Marks**
	DynamicArray.getSize	**01**
	DynamicArray.getCapacity	**01**
	DynamicArray.insert	**02**
	DynamicArray.search	**01**
	DynamicArray.erase	**01**
	DynamicArray.clear	**01**
	DynamicArray.operator++(prefix inrement)	**02**
	DynamicArray.operator++(postfix inrement)	**02**
	DynamicArray.operator--(prefix decrement)	**02**
	DynamicArray.operator--(postfix decrement)	**02**
	DynamicArray.operator+	**02**
	DynamicArray.operator-	**02**
	*DynamicArray.operator**	**02**
	DynamicArray.operator[]	**02**
	DynamicArray.operator>>	**02**
	DynamicArray.operator<<	**02**
	load() function	**02**
	save() function	**02**
	SimpleSet.default-constructor	**03**
	SimpleSet.regular-constructor	**03**
	SimpleSet.destructor	**00**
	SimpleSet.copy-constructor	**04**
	SimpleSet.move-constructor	**04**
	SimpleSet.copy-assignment	**04**
	SimpleSet.move-assignment	**04**
	SimpleSet.insert	**04**
	SimpleSet.operator+	**04**
	SimpleSet.operator-	**04**
	*SimpleSet.operator**	**04**
	SimpleSet.operator>>	**03**
	SimpleSet.operator<<	**03**
Makefile		**00**
README		**02**
Code Quality and Comments		**10**
Total		**100**