PROGRAMMING ASSIGNMENT 6

CS1410 - 100 points

OUTCOMES

After you finish this assignment, you will be able to do the following:

- Define an abstract class
- Create concrete classes from an abstract class
- Overload an operator
- Split classes into **`.h`** and **`.cpp`** files
- Open files for reading
- Write to files
- Use output manipulators such as **`setw`**, **`fixed`**, and **`setprecision`**

DESCRIPTION

A binary arithmetic operation takes two double operands (left and right) to perform addition, subtraction, multiplication, or division on. Write a program that reads basic arithmetic operation commands from a given input file named **`operations-in.txt`**, performs the operations, and saves their results to an output file named **`operations-out.txt`**. For example, if the input file has the following operation commands:

```
+  20  11
-  96  75
*  87  70
/  22   9
```

The program output file should be like this:

```
20.00 +    11.00 =      31.00
96.00 -    75.00 =      21.00
87.00 *    70.00 =    6090.00
22.00 /     9.00 =       2.44
```

You must follow the instructions below to create this program in an Object-Oriented Programming (OOP) way.

- In a header file named **`operation.h`**, define an **`abstract`** class named **`Operation`** with two **`protected double`** data members (**`left`** and **`right`**) and with the following public member functions:
    - **`Operation(double l, double r)`**: a two-argument constructor initializing the **`left`** and **`right`** data members of this class.

    - **`double perform() const`**: a pure **`virtual`** function that actually performs the operation (adds, substracts, multiplies, or divides) on the **`left`** and **`right`** data members and returns the result. This function must be implemented by all classes inheriting from this class.
    - **`char symbol() const`**: a pure **`virtual`** function returning the symbol character(**`+`**, **`-`**, **`*`**, or **`/`**) that represents the operation. This function must also be implemented by all classes inheriting from this class.
    - **`~Operation()`**: a **`virtual`** empty constructor.

  The **`Operation`** class should also define a **`friend`** function that overloads the **`<<`** operator. The prototype of this function should be like this:

  ```
  friend ostream& operator<<(ostream& out, const Operation& opr);
  ```

  This operator prints out the operation and its result to the output stream **`out`** in a format similar to that of the output file above. Use the **`setw`**, **`fixed`**, and **`setprecision`** manipulators to achieve that format.
- Define a class named **`Addition`** that inherits from the **`Operation`** class and represents the addition operation. This class must have a public two-argument constructor, delegating the initialization of the **`left`** and **`right`** data members to the constructor of the **`Operation`** class, and an empty destructor. It also must provide an implementation for the **`perform()`** and **`symbol()`** functions inherited from the **`Operation`** class. Split the code of this class into a header file name **`addition.h`** and an implementation file named **`addition.cpp`**.
- Repeat the previous step for the subtraction operation. The class should be named **`Subtraction`** and its code should be split into a header file name **`subtraction.h`** and an implementation file named **`subtraction.cpp`**.
- Repeat the previous step for the multiplication operation. The class should be named **`Multiplication`** and its code should be split into a header file name **`multiplication.h`** and an implementation file named **`multiplication.cpp`**.
- Repeat the previous step for the division operation. The class should be named **`Division`** and its code should be split into a header file name **`division.h`** and an implementation file named **`division.cpp`**.
- In a separate .cpp file, write a **`main()`** function that does the following
    - Open the given input file **`operations-in.txt`** for reading and read it one operation command at a time.
    - Open an output file named **`operations-out.txt`** for writing.
    - For each operation command read from the input operations-in.txt file do the following:
        - Based on the operation symbol, create a heap object of the appropriate class (Addition, Subtraction, Multiplication, and Division) based using the new operator
        - Use the << operator to print the operation and its result to the console.
        - Use the << operator to write the operation along with its result to the output file.
        - Delete the object when it's no longer needed.

- Close both input and output files

INSTRUCTIONS

For this assignment, you need to have a GitHub account. If you don't have one already, please sign up for one at https://github.com/.

Getting the assignment starter code from GitHub:

- Sign in to GitHub.
- Go to the assignment link https://classroom.github.com/a/DH15xpFo and accept the assignment. This should create a private repository under your GitHub username for this assignment. Click on the given link to open this repository and see the starter code.
- Click on the **Clone or Download** button dropdown and copy the given URL.
- Navigate to your assignments folder (or any folder you want this assignment to be placed in) and open it using Visual Studio code.
- In Visual Studio Code, open a new terminal and then run:

  `wsl` (for Windows 10 only)

  `git clone THE_URL_YOU_COPIED`

  This will download the starter code of this assignment from GitHub and create a folder for it with a name like **`cs1410-assignment-XX-github_username`**. This is the folder where your program file(s) (**`.cpp`** and/or **`.h`**) should reside.
- Open the assignment folder (whose name looks like **`cs1410-assignment-XX-github_username)`** in Visual Studio Code and start writing your program.

Compiling your C++ program:

- From inside the assignment folder in Visual Studio Code, open a new terminal and run:

  `wsl` (for Windows 10 only)
- To compile your program run:

  `make`

  This command will call the C++ compiler on your program, compile it, and, if no compilation errors are found, create an executable program named "**`run`**" for it. If there are compilation errors, read the console error messages and then go back to your source files (.cpp and/or .h) and fix them. Save your changes and run the "**`make`**" command to compile the program again.
- To run your program, run:

  `./run`
- To clean (remove) old compilation files and start over, run the command:

  `make clean`

  You can now run the "**`make`**" command to compile your program again and the "**`./run`**" command to run it.

Submitting your program to GitHub:

- Make sure to save your changes and commit them to GitHub when you are done. You can do that by running the following commands from inside your assignment folder:

  `wsl` (for Windows 10 only)

  `git add .`

  `git commit -m 'short commit message goes here'`

  `git push`

  Make sure to do this at least once by the deadline. For your final submission, I recommend using "**Final submission**" for the commit message. **Note that committing changes is not enough; you have to push them to GitHub; otherwise, your changes will stay on your local machine and I will not be able to see your submission.**
- Go to your assignment repository in github.com and make sure your changes are there.
- Click on the **Clone or Download** button dropdown and copy the given URL. Go to Canvas and submit the copied URL. **This URL must be submitted in Canvas after you make your "Final submission" to GitHub.**

RUBRIC

CRITERIA	POINTS
The **`Operation`** class	15
The **`Addition`** class	10
The **`Subtraction`** class	10
The **`Multiplication`** class	10
The **`Division`** class	10
Use of **`.h`** and **`.cpp`**	10
`main()`: Reading from and writing to files	19
Use of **`setw`**, **`fixed`**, and **`setprecision`**	6
Readable, commented, and properly indented code	10
TOTAL	100