CS2250 Spring 2018 Hauschild

PROGRAMMING ASSIGNMENT 5:
Due date: 4/25/2018

[70 points] A complex number is a number in the form of a + bi, where a and b are real numbers and i is the square root of negative 1.

Design and create a class called Complex for representing complex numbers. This class should have two attributes, a and b, which represent the parts of the complex number. This class should overload the +,-,*,/,++ and -- operators (both prefix and suffix) to perform basic complex number calculations according to the following formulas:
a + bi + c + di = (a+c) + (b+d)i
a + bi - (c + di) = (a-c) + (b - d)i
(a + bi)*(c + di) = (ac - bd) + (bc + ad)i
(a + bi)/(c + di) = (ac + bd)/(c^2 + d^2) + (bc - ad)i/(c^2 + d^2)
For example, 5 + 3i + 6 + 2i = 11 + 5i
(5+3i)*(6+2i) = (5*6 - 3*2) + (3*6 + 5*2)i = 24 + 28i
++ and -- should simply add or subtract one to the real part (a)

Provide three constructors (even if you do not use them all, I want them implemented), Complex (a,b), Complex (a) and Complex(). Complex() gives a number with a and b = 0, Complex(a) gives a number with a set but b=0, and Complex(a,b) sets both values. Your class should also have getRealPart() and getImaginaryPart() methods for returning the value of a and b respectively, as well as a method that returns the string representation of the number (to be used for displaying purposes later).

For your main program that will use the Complex class, prompt the user for how many complex numbers they wish to use. Then create an array of Complex numbers and prompt the user for values a and b for the entire array.

Now display the entire array and ask the user which number they would like to work with. Once the user has selected a number, ask the user if they would like to do a unary or binary operation. If they answer unary, prompt for ++prefix, postfix++,--prefix,postfix-- and then what number to store the result in. Then perform that operation. If they say they want to perform a binary operation, prompt for an operation (+,-,*,/) and then the second operand. Lastly, ask them in which number they wish to store the result. Perform the calculation and put the result where they told you to put it. In your code, use **overloaded operators** to do the work. This means you should implement overloaded operators of +,-,*,/,++,--(pre and post) and =(assignment).

So if users have the following list of numbers
1) 5 + 3i
2) 2 - 4i
3) 3 + 5i
4) quit
and they select 1,
then prompt them for
1) unary
2) binary
if they select 2,
then ask them what type of operation
1) +
2) -
3) *
4) /
if they enter +, then redisplay the numbers and prompt them for the second operand
1) 5 + 3i
2) 2 - 4i
3) 3 + 5i
4) quit
If they enter 2, then ask them for where to store the result. Suppose they enter 3 for the result. Then you want to do number 1 + number 2 and store the result in 3.
(5+3i) + (2 - 4i) and store the result in number 3

Your program should continue with this in a menu until the user selects quit.

[10] Exception handling: Your program should deal with the divide by zero error by throwing an exception in the divide overload, catching it in the main body and outputting an appropriate error message, after which you can let the program terminate.

Implementation: I strongly suggest you start by implementing the addition operator and testing it in a simple main body. Get that to work, then do subtraction and make sure it works. Get your operators tested and working before messing around with the logic of the menu and before working with the prefix and postfix operators.

If you are unsure about the mathematics here, talk to me. You don't need to know anything about complex numbers other than following the formula's given, so relax and don't get intimidated and if you are confused I can go over some examples with you.

This program will require 3 files.
p4.cpp or whatever you call the main program.
Complex.h: header file for Complex class
Complex.cpp: The file containing Complex's methods