Lab 10: Operator Overloading  
In this lab you will practice overloading the following C++ operators:

* addition (+)
* negation (unary -)
* insertion (<<)

Basic Structure  
In this lab, you will be implementing a class called Complex to represent complex numbers. A complex number consists of a real part and an imaginary part. It is represented as *x* + *y*i where *x* and *y* are the real and imaginary parts, respectively.

*Operator overloading* is a powerful feature of C++ that allows us to define operators so that they function with user-defined data types similarly to built-in data types. Just as we can add two doubles using "+", operator overloading allows us to add two Complexobjects with "+".

For this lab, you will overload the binary addition operator (+), the unary negation (-), and the output operator << such that

* binary + allows us to add two complex numbers,
* unary - allows us to negate a complex number (both the real and imaginary parts are negated), and
* << allows us to display complex numbers.

To specify the behavior of an overloaded operator, we need to write an operator function. The following is the prototype of such a function:   
  
*return-type* operator *op* (*parameters*);   
  
where *return-type* is the type of value returned, *op* is the operator being overloaded, and *parameters* are the parameters of the function.

The operator functions can be either member or non-member functions. We will be using both options in this lab.

Lab Set-up  
If you've ssh’ed into the GL servers, you can copy the lab files to your directory with the following commands:

cp /afs/umbc.edu/users/c/m/cmarron/pub/cmsc202/lab10.zip .

unzip lab10.zip

(Note the '.' at the end of the first command – it is very important.)

After you unzip lab10.zip, you will have the following files in your directory:

1. [Complex.h](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab10/Complex.h): The Complex class is defined in this file.
2. [Complex.cpp](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab10/Complex.cpp): You will implement the operator functions in this file. The constructor and accessors are provided.
3. [lab10.cpp](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab10/lab10.cpp): This is the driver program provided to you; you do **NOT** need to modify this. It asks the user to input two complex numbers and then displays the negation of each complex number along with their sum.

The Complex Class

The Complex class will represent complex numbers. The file Complex.h contains the definition of the Complex class. The private data members and public methods are already defined. Complex.cpp should consist of following:

1. Constructor: the constructor for the Complex class should accept the real and imaginary parts of a complex number and store them in the appropriate data members.
2. GetReal(): an accessor that returns the real part of the complex number.
3. GetImaginary(): an accessor that returns the imaginary part of the complex number.
4. operator +(): adds two complex numbers and returns the result. Since addition of two complex numbers is also a complex number, it returns an object of typeComplex. More details are provided later in the lab.
5. operator -(): negates the real and imaginary parts and returns the result. Since negation of a complex number is also a complex number, it returns an object of type Complex. More details are provided later in the lab.

In addition to the above, the code for overloading the insertion operator (<<) must also be written in Complex.cpp. However, it is a non-member function and so is not part of the Complex class. More details are provided later in the lab.

Overloading the "+" Operator  
We will overload the + operator so that it will add two Complex objects and return the result as a Complex object. As we have learned in the lectures on overloading C++ operators, the operator function for + can either be a *member function* or a *non-member function*. For the purpose of this lab, we will implement this operator function as a member function.

The operator + is a binary operator (meaning it takes two operands). The prototype of the function when implemented as member function is:   
  
const Complex operator+ (const Complex & rhs) const;   
  
The operator+ function is implemented in the following manner:

* If you are adding two Complex objects C1 and C2 to get another Complex object C3= C1 + C2, then C1 is the calling object (this), and rhs refers to the right-hand side (C2) of the expression C1 + C2. The statement C3 = C1 + C2; invokes theoperator+ function, and is equivalent to C3 = C1.operator+(C2).
* The the left-hand side (C1) is passed implicitly. Hence we only need to explicitly pass rhs. This is because we are implementing operator+ as a member function. If we wrote it as non-member function, we would need to pass both the left-hand and right-hand sides explicitly.
* Create a Complex object in this function (C3 in the example) and assign the sum of the real parts of C1 and C2 to the real part of C3. Similarly, assign the sum of the imaginary parts of C1 and C2 to the imaginary part of C3.
* Return the object C3. Note that the return type is by const value.

Overloading the Unary "-" Operator  
We will overload the unary minus operator so that it will negate a Complex object. A unary minus operator - has only one operand. The prototype of the function when implemented as a member function is:   
  
const Complex operator- () const;   
  
The operator- function is implemented in the following manner:

* The function will perform the negation as C3 = -C1.
* There is only one operand (C1) which is passed implicitly. Hence we do not need to pass anything to operator-.
* Create a Complex object in this function (C3 in the example) and assign the negation of of the real part of C1 to the real part of C3. Similarly, assign the negation of the imaginary part of C1 to the imaginary part of C3.
* Return the object C3. Note that the return type is by const value.

Overloading the "<<" Operator  
The insertion operator << is the most commonly overloaded operator. Just as we usecout to display values of basic data types, we will overload << so that it displays ourComplex objects. The strange thing about the << operator is that is always operates on a heterogeneous pair of operands: the left-hand side is cout or some other ostream, and the right-hand side is the class you are trying to output. When an operator function is overloaded as a member function, the left-hand side calling object is passed to the function implicitly; for the insertion operator, that would be cout, which is not a class we can add a member function to. So, the operator << is not overloaded as a member function. Hence it is not a part of our Complex class. However, we do define it inComplex.h and implement in Complex.cpp.

The operator << is a binary operator: the left-hand operand is the ostream object coutand the right-hand operand is our Complex object. The prototype of the operator function for overloading << is:   
  
ostream& operator<< (ostream& out, const Complex& number);   
  
The operator << can also be implemented as a friend function. Since we have accessors for our Complex class (GetReal() and GetImaginary()), there is no need to make it a friend function.

Let number refer to the Complex object we wish to display The operator<< function is implemented in the following manner:

* The left-hand side is cout, which is passed explicitly, as operator<< is a non-member function.
* You should use the accessors of the Complex class, GetReal() and GetImaginary(), to get the real part and imaginary part of number and display them. Note that the output should be in the form:   
    
  Real\_part +/- Imaginary\_part i
* There is a small complication in outputting the imaginary part of complex numbers: if it is non-negative, you must output a "+" between the real and imaginary parts, and if it is negative, you must output a "-". For example, the following are correct outputs: 3 + 2i, -3 + 2i, 3 - 2i, -3 - 2i. But this is an incorrect output: 3 + -2i (it should be 3 - 2i).

The lab10.cpp Driver

The file lab10.cpp is the driver program provided to you. You do **NOT** need to modify this file. This program accepts two complex numbers from the user. Then it displays the negation of each number and the sum of the two numbers. It uses overloaded << operator to produce the ouptut.

# Compiling and Running Compiling  g++ -Wall -ansi lab10.cpp Complex.cpp -o lab10  Run  linux3[22]% ./lab10  Sample output linux3[20]% ./lab10 Enter the first complex number :  Real Part : 2  Imaginary Part : 3  Enter the second complex number :  Real Part : 5  Imaginary Part : -4  The Complex numbers you entered are :  First number : 2 + 3i  Second number : 5 - 4i  The negation of first number is : -2 - 3i  The negation of second number is : -5 + 4i  The addition of the two numbers is : 7 - 1i  linux3[21]% ./lab10 Enter the first complex number :  Real Part : 0  Imaginary Part : 3  Enter the second complex number :  Real Part : -2  Imaginary Part : 5  The Complex numbers you entered are :  First number : 0 + 3i  Second number : -2 + 5i  The negation of first number is : 0 - 3i  The negation of second number is : 2 - 5i  The addition of the two numbers is : -2 + 8i