1 Classes and Data Abstraction II, Lab Exercise 2 - Complex Numbers Again

Create a class called **Complex** for performing arithmetic with complex numbers. Write a driver program to test your class. Complex numbers have the form

$$realpart + imaginarypart * i$$

where **i** is

$$\sqrt{-1}$$

Use floating–point variables to represent the **private** data of the class. Provide a constructor function that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide **public** member functions for each of the following:

- Addition of two **Complex** numbers: The real parts are added together and the imaginary parts are added together
- Substraction of two **Complex** numbers: The real part of the right operand is subtracted from the real part of the left operand and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.
- Printing **Complex** numbers in the form **(a,b)** where **a** is the real part and **b** is the imaginary part.

  The output should appear as follows:

```
(1,7) + (9,2) = (10,9)
(10,1) - (11,5) = (-1,-4)
```

```
// complex.h
#ifndef COMPLEX_H
#define COMPLEX_H
/* Write class definition for Complex */
#endif
-----------------------------------------------
// complexM.cpp
// member function definitions for class Complex
#include <iostream>
using std::cout;
#include "complex.h"
Complex::Complex( double real, double imaginary )
  { setComplexNumber( real, imaginary ); }
```

```
void Complex::addition( const Complex &a )
{
   /* Write statement to add the realPart of a to the class
      realPart */
   /* Write statement to add the imaginaryPart of a to the
      class imaginaryPart */
}
void Complex::subtraction( const Complex &s )
{
   /* Write a statement to subtract the realPart of s from the
      class realPart */
   /* Write a statement to subtract the imaginaryPart of s from
      the class imaginaryPart */
}
void Complex::printComplex( void )
  { cout << '(' << realPart << ", " << imaginaryPart << ')'; }
void Complex::setComplexNumber( double real, double imaginary )
{
   realPart = real;
   imaginaryPart = imaginary;
}
-----------------------------------------------
// complex.cpp
#include <iostream>
using std::cout; using std::endl;
#include "complex.h"
int main()
{
   Complex b( 1, 7 ), c( 9, 2 );
   b.printComplex();
   cout << " + ";
   c.printComplex();
   cout << " = ";
   b.addition( c );
   b.printComplex();
   cout << '\n';
   b.setComplexNumber( 10, 1 );
   c.setComplexNumber( 11, 5 );
   b.printComplex();
   cout << " - ";
   c.printComplex();
   cout << " = ";
   b.subtraction( c );
```

```
   b.printComplex();
   cout << endl;
   return 0;
}
-------------------------------------------------
```

Tips:

- You must write the definition for class **Complex**. Use the details provided in the member function definitions (complexM.cpp) to assist you.
- Remember to use member–access specifiers **public** and **private** to specify the access level of data members and functions. Carefully consider which access specifier to use for each class member. In general, data members should be **private** and member functions should be **public**.

Questions

1. Why do you think **const** was used in the parameter list of **addition** and **subtraction**?
2. Can **addition** and **subtraction**'s parameters be passed call–by–value instead of call–by–reference? How might this affect the design of class **Complex**? Write a new class definition.
3. Declare a **Complex** number as follows, without passing any arguments to the constructor. What happens? Does the default constructor handle this declaration?
   Complex a;