Homework #4 – Class Complex

In this assignment you are asked to implement a `Complex` class that represents a complex number. A complex number consists of a real part + an imaginary part `* i` where `i` has the value `sqrt(-1)`. You can learn more about complex numbers at http://en.wikipedia.org/wiki/Complex_number. The following UML class diagram shows the attributes and behaviors of class `Complex`. *Note: UML type Float will map to type `double` when implemented in C++.*

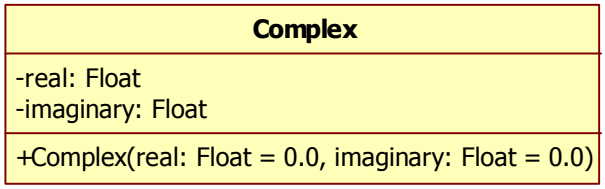

Figure 1. UML class diagram for class Complex

1. **(1 point)** Make sure the basic requirements are met:
   a. Create files named "Complex.h" and "Complex.cpp" to hold your implementation of class `Complex`.
   b. Define a namespace based on your first and last name (e.g. "RayMitchell") in which you will define class `Complex`.
   c. Implement class `Complex` as shown in the UML diagram.
2. **(3 points)** Use **friend** functions to overload the stream insertion (<<) and extraction (>>) operators for class `Complex`. Both operators must work with fully-formatted complex numbers (e.g. "1+2i"). The stream insertion operator must output complex numbers in this format; the stream extraction operator must read complex numbers in this format (*Hint: You can extract a single character from an `istream` by extracting into a variable of type `char`*). The stream extraction operator does not need to handle invalid input or whitespace embedded within the complex number being read.
3. **(3 points)** Use **member** functions to overload the addition (+), subtraction (-), equivalence (==), and non-equivalence (!=) operators for class `Complex`.
4. **(1 point)** Make sure `const` is used correctly throughout class `Complex`. Be sure to check all pointer parameters, reference parameters, and member functions for proper "const-ness".
5. **(1 point)** Write a test program that demonstrates class `Complex`'s capabilities. Your test program should demonstrate all constructors, public member functions, and friend functions. It should also verify that all error conditions (if any) are handled properly. Place your test program in a file named "hw4.cpp".
6. **(1 point)** Make sure your source code is well-commented, consistently formatted, uses no magic numbers/values, follows a consistent style, and is ANSI-compliant.

Place all source code and a screen capture of the output produced by your program in a single Word or PDF document. Submit this document.