Practice questions for week 6

Consider class `Complex`. The class enables operations on so-called complex numbers. These are numbers of the form `realPart + imaginaryPart *` i, where i has the value

$$\sqrt{-1}$$

a. Modify the class to enable input and output of complex numbers through the overloaded >> and << operators, respectively (you should remove the print function from the class).
b. Overload the multiplication operator to enable multiplication of two complex numbers as in algebra.

Overload the == and != operators to allow comparisons of complex numbers.

A machine with 32-bit integers can represent integers in the range of approximately 2 billion to +2 billion. This fixed-size restriction is rarely troublesome, but there are applications in which we would like to be able to use a much wider range of integers. This is what C++ was built to do, namely, create powerful new data types. Consider class `HugeInt` mentioned in Deitel's book. Study the class carefully, then answer the following:

a. Describe precisely how it operates.
b. What restrictions does the class have?
c. Overload the * multiplication operator.
d. Overload the / division operator.
e. Overload all the relational and equality operators.

[Note: We do not show an assignment operator or copy constructor for class `HugeInteger`, because the assignment operator and copy constructor provided by the compiler are capable of copying the entire array data member properly.]

Create a class `RationalNumber` (fractions) with the following capabilities:

a. Create a constructor that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators.
b. Overload the addition, subtraction, multiplication and division operators for this class.

Overload the relational and equality operators for this class.

Develop class `Polynomial`. The internal representation of a `Polynomial` is an array of terms. Each term contains a coefficient and an exponent. The term

$2x^4$

has the coefficient 2 and the exponent 4. Develop a complete class containing proper constructor and destructor functions as well as set and get functions. The class should also provide the following overloaded operator capabilities:

a. Overload the addition operator (`+`) to add two `Polynomial`s.
b. Overload the subtraction operator (`-`) to subtract two `Polynomial`s.
c. Overload the assignment operator to assign one `Polynomial` to another.
d. Overload the multiplication operator (`*`) to multiply two `Polynomial`s.

Overload the addition assignment operator (`+=`), subtraction assignment operator (`-=`), and multiplication assignment operator (`*=`).