Practice questions for week 4

(`Complex` Class) Create a class called `Complex` for performing arithmetic with complex numbers. Write a program to test your class.

Complex numbers have the form

```
realPart + imaginaryPart * i
```

where i is $\sqrt{-1}$

Use `double` variables to represent the `private` data of the class. Provide a constructor 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 that perform the following tasks:

a. Adding two `Complex` numbers: The real parts are added together and the imaginary parts are added together.
b. Subtracting 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.
c. Printing `Complex` numbers in the form (`a, b`), where `a` is the real part and `b` is the imaginary part.

(`Rational` Class) Create a class called `Rational` for performing arithmetic with fractions. Write a program to test your class.

Use integer variables to represent the `private` data of the classthe `numerator` and the `denominator`. Provide a constructor 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 and should store the fraction in reduced form. For example, the fraction

$$\frac{2}{4}$$

would be stored in the object as 1 in the `numerator` and 2 in the `denominator`. Provide `public` member functions that perform each of the following tasks:

a. Adding two `Rational` numbers. The result should be stored in reduced form.
b. Subtracting two `Rational` numbers. The result should be stored in reduced form.
c. Multiplying two `Rational` numbers. The result should be stored in reduced form.
d. Dividing two `Rational` numbers. The result should be stored in reduced form.
e. Printing `Rational` numbers in the form `a/b`, where `a` is the numerator and `b` is the denominator.

f. Printing `Rational` numbers in floating-point format.

(`HugeInteger` Class) Create a class `HugeInteger` that uses a 40-element array of digits to store integers as large as 40 digits each. Provide member functions `input`, `output`, `add` and `substract`. For comparing `HugeInteger` objects, provide functions `isEqualTo`, `isNotEqualTo`, `isGreaterThan`, `isLessThan`, `isGreaterThanOrEqualTo` and `isLessThanOrEqualTo`each of these is a "predicate" function that simply returns `TRue` if the relationship holds between the two `HugeIntegers` and returns `false` if the relationship does not hold. Also, provide a predicate function `isZero`. If you feel ambitious, provide member functions `multiply`, `divide` and `modulus`.

(`TicTacToe` Class) Create a class `TicTacToe` that will enable you to write a complete program to play the game of tic-tac-toe. The class contains as `private` data a 3-by-3 two-dimensional array of integers. The constructor should initialize the empty board to all zeros. Allow two human players. Wherever the first player moves, place a 1 in the specified square. Place a 2 wherever the second player moves. Each move must be to an empty square. After each move, determine whether the game has been won or is a draw. If you feel ambitious, modify your program so that the computer makes the moves for one of the players. Also, allow the player to specify whether he or she wants to go first or second. If you feel exceptionally ambitious, develop a program that will play three-dimensional tic-tac-toe on a 4-by-4-by-4 board. [Caution: This is an extremely challenging project that could take many weeks of effort!]