OOP Homework Assignment # 1 [50 points]

RELEASE DATE: 26TH MARCH 2010
DUE DATE: 9TH APRIL 2010

Question 1: Complex Number Class [10]

Following is a declaration for a class to represent Complex numbers. A Complex number has two parts, the real part (let's say a) and the imaginary part(say b), and is represented as a+bi where i has a value of sqrt(-1). **Write the implementation of the class**. **Also add overloaded operators to add, subtract, multiply and divide two complex numbers.**

```
class complex
{
     private:
        float real;       // Real Part
        float imag;      //  Imaginary Part

   public:
      complex(); //constructor to set real and imag to 0.0
      complex(float,float); //2-arg constructor
      complex(complex&); //copy constructor

      complex add(complex&);
      complex subtract(complex&);
      complex multiply(complex&);
      complex divide(complex&);
      complex getconjugate();

      void setdata(float,float); //assigns the values passed as   arguments to the
      object on which setdata is called

      void getdata(); //takes real and imag as input from user
      float getreal();  //returns data member real
      float getimaginary(); //returns data member imag
      void display(); //displays the complex number in the form a+bi
 };
```

Addition of two complex numbers

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Multiplication of two complex numbers

$$(a + bi)(c + di) = (ac - bd) + (bc + ad)i$$

Subtraction of two complex numbers

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

Division of two complex numbers

$$\frac{a+bi}{c+di} = \left(\frac{ac+bd}{c^2+d^2}\right) + \left(\frac{bc-ad}{c^2+d^2}\right)$$

Conjugate of a complex number

The conjugate of a + bi is a -bi

Question 2: TicTacToe Application [25]

Create a class TicTacToe that will enable you to write a complete program to play the game of Tic-Tac-Toe. The class contains a private 3-by-3 two-dimensional array of integers. The constructor should initialize the empty board to all zeros. Allow two human players. Whenever the first player moves, place a 1 in the specified square, and 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 and whether it is a draw. The details are given below:

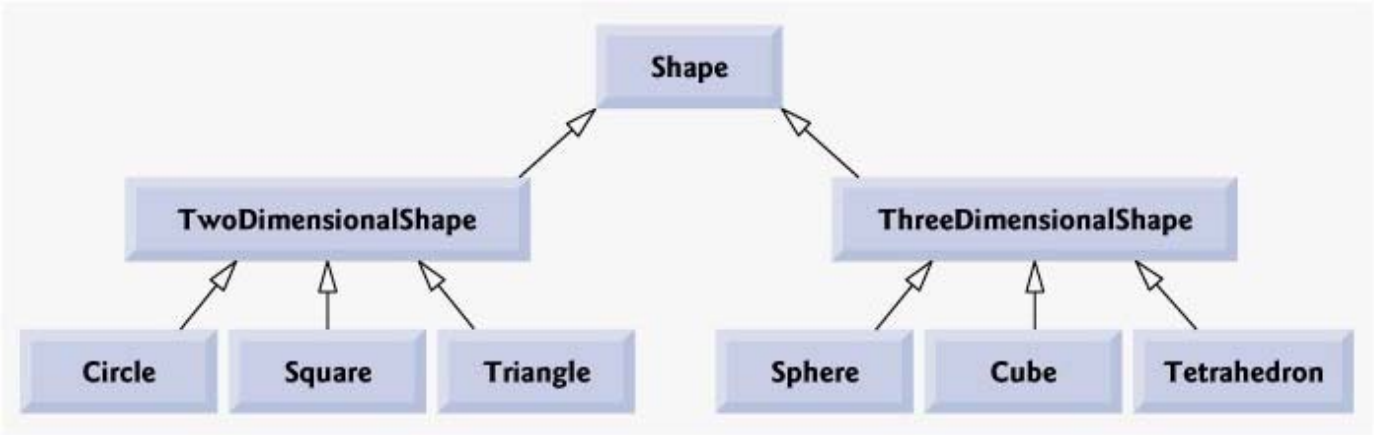
- The class should contain as its private data a 3 by 3 array of integers called board.
- A constructor should initialize the empty board to all zeroes.
- A method called P1_move() will place a 1 in the specified position. This method shall take the position as an argument and returns nothing.
- A method called P2_move() will place a 2 in the specified position. This method shall take the position as an argument and returns nothing.
- Make sure that each move must be to an empty square.
- A method called EndGame() should determine after each move whether the game has been won or is a draw. This function should return a char value to decide whether to end the game or continue.
- A method called askNumber() should ask the user to enter the number and generate an error message if the number is out of range (not between 0 and 8 – *look below to see how you are going to display your board*). It should implement the logic to keep asking the user for a number until he enters a correct number. This function will return the number entered by the user.
- A displayBoard() Function should display the updated board to the players after each move in the same format as given in the instructions function below.
- A non-member function called instructions () is implemented below which will be called directly in main at the start of the program.

```
void instructions() //display instructions
{
cout << "Welcome to Tic-Tac-Toe.\n";
cout << "Make your move known by entering a number, 0 - 8. The number\n";
cout << "corresponds to the desired board position, as illustrated:\n\n";

cout << "0 | 1 | 2\n";
cout << "---------\n";
cout << "3 | 4 | 5\n";
cout << "---------\n";
cout << "6 | 7 | 8\n\n";

}
```

Note: You may want to write some more functions in the class in order to make the code easier to write. For example, you may add a function validMove() to check whether a move to a certain position is valid or not, a board_full() function to check whether the board is full or not. You might want to add a function to see if there is any winner or if the game is withdrawn. These are all optional and only if you want to add them to your code.

Question 3: Shape Hierarchy [7]

Implement the shape hierarchy shown in the following figure. Each TwoDimensionalShape should contain method calcArea to calculate the area of the two dimensional shape. Each ThreeDimensionalShape should have methods calcArea and calcVolume to calculate the surface area and volume, respectively, of the three-dimensional shape. (You might have to google to find out the formulae for finding areas of each of these shapes!) In main(), create shapes of different types and display their area, and in case of 3D shapes, the volume as well.



Question 4: `Package` Inheritance Hierarchy [8]

Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use `Package` as the base class of the hierarchy, then include classes `TwoDayPackage` and `OvernightPackage` that derive from `Package`. Base class `Package` should include data members representing the name, address and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. `Package`'s constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. `Package` should provide a `public` member function `calculateCost` that returns a `double` indicating the cost associated with shipping the package. `Package`'s `calculateCost` function should determine the cost by multiplying the weight by the cost per ounce. Derived class `TwoDayPackage` should inherit the functionality of base class `Package`, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. `TwoDayPackage`'s constructor should receive a value to initialize this data member. `TwoDayPackage` should redefine member function `calculateCost` so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class `Package`'s `calculateCost` function. Class `OvernightPackage` should inherit directly from class `Package` and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. `OvernightPackage` should redefine member function `calculateCost` so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a test program that creates objects of each type of `Package` and tests member function `calculateCost`.