Revision Lab

Question:

The courier service TCS offers a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. In this implementation, we will use **Package** as the base (abstract) class of the hierarchy and derive two classes called **TwoDayPackage** and **OvernightPackage**.

1. Base class **Package** should include data members representing the **name**, **address**, **city**, and **phoneNumber** for both the sender and the recipient of the package (total 8 data members). Use string and int data members. Additionally, two more float data members store the **weight** (in ounces) and **cost** per ounce to ship the package. Ensure that creating object of **Package** class is not allowed in main program.

   The **package's** constructor should initialize all of these data members (no setters or getters required). Assume that the **weight** and **cost** per ounce contain positive values. **Package** should provide a pure virtual member function **calculateCost** that returns a double indicating the cost associated with shipping the package. (Checklist: Total 10 data members, one pure virtual function, and one parametarized constructor)

2. Derived class **TwoDayPackage** should inherit the functionality of base class **Package**, but additionally also include a float data member that represents a fixed **fee** that TCS charges for two-day-delivery service. **TwoDayPackage's** constructor should receive a value to initialize this data member. The inherited members should be initialized by calling parametarized constructor of **Package** class. **TwoDayPackage** should redefine member function **calculateCost** so that it computes the shipping cost. **calculateCost** function should determine the cost by multiplying the weight by the cost per ounce and the the fixed fee will be added to the weight-based cost.

3. Class **OvernightPackage** should inherit directly from class **Package** and contain an additional float 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 returning the total shipping cost.

4. Write a test program that creates objects of each type of Packages (overnight and two-day delivery) and tests member function **calculateCost**.

Question:

The al-Baraka bank offers the employees of UCP a loan for homes. However, the loan is subject to tax deductions as per Government rules. Create an inheritance hierarchy to represent two types of loan offers (1) forFiler (2) forNonFiler. In this implementation, we will use **Loan** as the base (abstract) class of the hierarchy and derive two classes called **ForFiler** and **ForNonFiler**.

1. Base class **Loan** should include data members representing the **name**, **address**, **city**, **phoneNumber,** and **grossSalary**. Use string and int data members. Additionally, one more float

data member stores the **taxRate** (in percentage i.e. 1.5 means 1.5%)**.** Ensure that creating objects of **Loan** class is not allowed in the main program.

The **Loan's** constructor should initialize all of these data members (no setters or getters required). Assume that the **grossSalary** and **taxRate** have positive values. **Loan** should provide a pure virtual member function **calculateTax** that returns a double indicating the total payable tax. (Checklist: Total 6 data members, one pure virtual function, and one parametarized constructor).

2. Derived class **ForFiler** should inherit the functionality of base class **Loan. ForFiler's** parametraized constructor should receive all values and the inherited members should be initialized by calling parametarized constructor of **Loan** class. **ForFiler** should redefine member function **calculateTax** so that it computes the payable tax. **calculateTax** function should determine the tax by multiplying the **grossSalary** by the **taxRate**. The taxRate for filer is 1.5% of the gross salary.

3. Class **ForNonFiler** should inherit directly from class **Loan** and contain an additional float data member-representing **penalty** charged by the Government. **ForNonFiler** should redefine member function **calculateTax** so that it adds the **penalty** to the standard tax calculated by multiplying the **grossSalary** by the **taxRate** before returning the total tax. The tax for non-filer is 2.5% of the gross salary and the penalty is an additional amount of 0.5% of the gross salary.

4. Write a test program that creates objects of each type of Loan (**ForFiler** and **ForNonFiler**) and tests member function **calculateTax**.

Question:

In this question, you will develop a class hierarchy of shapes and write a program that computes the amount of paint needed to paint different objects. The hierarchy will consist of a parent class Shape with three derived classes - Sphere, Rectangle, and Cylinder. For the purposes of this exercise, the only attribute a shape will have is a name and the method of interest will be one that computes the area of the shape (surface area in the case of three-dimensional shapes).

Do the following.

a. Write an abstract class Shape with the following properties. An instance variable shapeName of type char*, An abstract method area() , A toString method that returns the name of the shape
b. Class for a sphere which is a descendant of Shape. A sphere has a radius and its area (surface area) is given by the formula 4*PI*radius^2. Define similar classes for a rectangle and a cylinder. Both the Rectangle class and the Cylinder class are descendants of the Shape class. A rectangle is defined by its length and width and its area is length times width. A cylinder is defined by a radius and height and its area (surface area) is PI*radius^2*height. Define the toString method in a way similar to that for the Sphere class.
c. Class for a type of paint (which has a "coverage" and a method to compute the amount of paint needed to paint a shape). Correct the return statement in the amount method so the correct amount will be returned. Use the fact that the amount of paint needed is

the area of the shape divided by the coverage for the paint. (NOTE: Leave the print statement - it is there for illustration purposes, so you can see the method operating on different types of Shape objects.)

d. In main a paint object has been instantiated. Add the following to complete the program. Instantiate the three shape objects: deck to be a 20 by 35 foot rectangle, bigBall to be a sphere of radius 15, and tank to be a cylinder of radius 10 and height 30. Make the appropriate method calls to assign the correct values to the three amount variables. Run the program and test it. You should see polymorphism in action as the amount method computes the amount of paint for various shapes.

e. At the end write the cout statements in constructors and destructors in Shape and its descendant classes after that in main delete the Base class pointer(i.e Shape *s) and see if there is any undefined behavior.

Question:

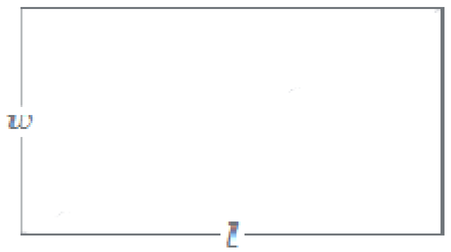
In geometry, a quadrilateral can be defined as a closed, two-dimensional shape, which has four straight lines. We can find the shape of quadrilaterals in various things around us, like in a chess board, a deck of cards, a kite, a sign board and in an arrow. Task is to implement this scenario in C++ using various OOP concepts as explained ahead.

Write a program that has an abstract base class named Quad. This class should have four data members representing sides (Data type of each side is 'Line', where 'Line' is also a class).

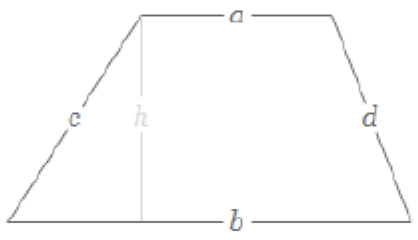
Also provide pure virtual functions to calculate Area and Perimeter respectively. It should also have a method for setting and getting the data variables.

Line class has an attribute to save the length of a side, parametrized constructor, setters & getters functions.

Derive a class Rectangle and Trapezoid (with addition float attribute named height) from Quad and override the Area and Perimeter method so that it returns the area and perimeter of the respected class.



Rectangle



Trapezoid

	Area	Perimeter
Rectangle	$\ell * w$	$2(\ell + w)$
Trapezoid	$\frac{h(a+b)}{2}$	$a + b + c + d$

Sample Main:

```
int main ()
{
        Quad *quadptr[2];
        quadptr[0] = new Rectangle (2,3) ;           // 2 is length, 3 is width
        quadptr[1] = new Trapezoid (5, 7, 3, 4, 6); // 5 is height, then side a,b,c,d respectively.

        for(int i =0; i<2; i++)
        {
              cout << quadptr[i]->area () << endl;         //return area of rectangle
              cout << quadptr[i]->perimeter () << endl; // return perimeter of rectangle
        }

        for(int i =0; i<2; i++)
             delete quadptr[i];           //Destructor should display the name of respective class

        return 0;
}
```