Instructions

Camel case notation

- void setAttributeName()

Lab 12 – Inheritance

Question 1

Write a class Date that has three protected data members of type int: day, month and year and it returns the date in the format 11/4/2017.

- Date(int day,int month,int year)
- string getDate() // format 11/4/2017.

Also write a Time class that has hours, minutes and seconds protected data members of type int. The Time class should returns the Time in following format 8:35:40

- string getTime() //format 8:35:40
- Time(int hours, int minutes, int seconds)

Now derive DateTime class from these two classes to display current date and time as follows: DateTime class should have a function

- DateTime(int day, int month, int year, int hours,int mins,int sec)
- string getDT()//return string of date and time

In this getDT() function you will be using functions parent class functions to produce desired output/string.

```
1/1/1900 0:0:0
4/2/1960 5:32:27
```

Question 2

Design a class named Person should have following protected data members.

- Name(string)
- Address(string)

The Employee class derived from Class Person should keep the following information as protected member variables:

- Employee number(int)
- Hire date(Date)

Write one or more constructors and the appropriate accessor and mutator functions for the class. Next, write a class named ProductionWorker that is derived from the Employee class. The ProductionWorker class should have member variables to hold the following information:

- Shift (an integer)
- Hourly pay rate (a double)

The workday is divided into two shifts: day and night. The shift variable will hold an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. Write one or more constructors and the appropriate accessor and mutator functions for the class by using camel case notation described in instructions.

b) In this part you will extend ProductionWorker class:
Assume that in a particular factory, a team leader is an hourly paid production worker who leads a small team. In addition to hourly pay, team leaders earn a fixed monthly bonus. Team leaders are required to attend a minimum number of hours of training per year. Design a TeamLeader class that extends the ProductionWorker class you designed above. The TeamLeader class should have member variables for the monthly bonus amount, the required number of training hours, and the number of training hours that the team leader has attended. Write one or more constructors and the appropriate accessor and mutator functions for the class and a function to calculate salary.

- double calculateSalary(Date dt)

you will receive a date object and from that date object you will calculate the no of days he/she worked in that particular month, you may assume team leader works 8 hours a day. Add monthly bonus according to days.

Calculate the salary of production worker in by creating an object in main.Don't include this main function while submitting.

Question 3

Write C++ Vertex class. In addition to having two integer member variables, x and y, the class should have two methods. A method for setting the x-y values to a random number in a given range(0-100), you should have proper getters and setters with constructors, follow camel casing notation, and a method for returning a string representation of the vertex.

- string stringRep()//will return string having info of x and y of vertex

Write a program defining a new class, SegList, which primarily holds a list of line segments based on list of vertices in the x-y plane. Your SegList class should have an array of Vertex objects(fixed size of 10), and be defined with a method for accepting new vertices and a method for producing a string representation the entire set of vertices.

- void addVertex(Vertex)//will add vertex to the array

- string getSpec()//will return string representation of vertices

This method for producing a string should utilize the method defined on the Vertex class for producing a string representation of the single vertex.

Create two subclasses of the SegList class called Triangle and Square. For now, these two classes will do nothing more than overload the getSpec() function.

- string getSpec()//will return a string having type and specs

The overloaded function should invoke the getSpec() function already implemented in the SegList superclass. The overloaded function should simply add the type to the output of the string spec.

In this exercise we will test the two new classes with a simple main() function that creates a Triangle instance and Square instance with vertices:

- (0,0), (10,0), and (5,5)
- (0,0), (10,0), (10,10), and (0,10)

After instantiating and populating two instances, with the above vertices, just output the result of getSpec() to the terminal. Don't forget to remove this main function while submitting as we will be using test case's main function.

```
Triangle: type=triangle,x=0,y=0,x=10,y=0,x=5,y=5
Square:   type=square,x=0,y=0,x=10,y=0,x=10,y=10,x=0,y=10
```