Object Oriented Programming Lab Spring 2019
Inheritance and Polymorphism
Lab 13

Submission Instructions

- Copy exact prototype from submission.cpp.
- Submit all header files and cpp files.

Before attempting the task read the concepts discussed below

>Abstract Classes:
An **abstract class** is a class that is designed to be specifically used as a base class. An abstract class contains **at least one pure virtual function**. You declare a pure virtual function by using a pure specifier (= 0) in the declaration of a virtual member function in the class declaration.

Example:

```
class AbstractClass {
public:
  virtual void AbstractMemberFunction() = 0; // Pure virtual function
                                            // makes this class Abstract class.

  virtual void NonAbstractMemberFunction1(); // Virtual function.

  void NonAbstractMemberFunction2();
};
```

The main difference between '**virtual function**' and '**pure virtual function**' is that 'virtual function' **has its definition in the base class** and also the inheriting derived classes redefine it. The pure virtual function **has no definition in the base class**, and all the inheriting derived classes have to redefine it.

>Be Careful !
Abstract class **cannot** be used as a parameter type, a function return type, and **not** to declare an object of an abstract class. It **can be** used to declare pointers and references to an abstract class.

>For initialization of base class members: Constructor of base class can only be called using Initializer List

Task# 1

Write a program to calculate the area of following shapes by using ***Public -- Single Inheritance***. The *base class* is "shape" and the *derived classes* are **rectangle**, **triangle** and **circle**. Attributes of all the classes are as under:

shape	rectangle	triangle	circle

protected:
string type;
public:
virtual void area ()=0;
 	
//area( )=0 is a pure virtual function, so we do not need to create a shape.cpp for its implementation
	public:
	void area ()
	{
	//definition
	}
	private:
	float height;
	float width;
		public:
		void area ()
		{
		//definition
		}
		private:
		float base;
		float height;
			public:
			void area ()
			{
			//definition
			}
			private:
			float radius;

	Area = width* height	Area = 1/2 of the base X the height	A = πr^2

- Your classes must have default constructor and parameterized constructor (see submission file)
- Provide a pure virtual display function in Shape
- Provide implementation of display function for all classes, in Shape Class, as the function Display the value of type as "Shape". In Rectangle the Display function should display

```
cout<<"Type : "<<type;
cout<<"Width :"<<width;
cout<<"Height :"<<height;
```

- Similarly provide the implementation of function display for all rest of classes according to their member functions.
- Since shape class is abstract and cannot be instantiated, but we can create a pointer of it and make it point to the objects of child classes' one by one, i.e.
  Shape* ptr=new **Rectangle** ("Rectangle", 4, 6); *(pointer of parent, object of child)*
- Similarly instantiate all child classes.
- Now call the area function for each child class to compute area.
- Call the display function as well.

Copy function headers from submission.cpp

Task# 2

Multilevel Inheritance Overriding

Multiple inheritances enable a derived class to inherit members from more than one parent. Here base classes are **Person** and **Employee**, Derived class is **Faculty**. Attributes are as under:

Person (Base Class)	Employee (derived from Person)	Faculty (Derived from Employee)

protected:
`string name;`
`string address;`
	protected:
	`int Emp_no;`
	`float gross_pay;`
	`float house_rent;`
	`float medical_allow;`
	`float net_pay;`
	
	`public:`
	virtual void calcSalary( )=0
		protected:
		`string designation;`
		`string department;`
		`string course;`
		
		`public:`
		virtual void calcSalary( )

Use the formula below to calculate ne_pay::

- *House rent is 45%.*
- *Medical Allowance is 5%.*

*Formula to calculate net_pay= gross_pay – ((45/100)*gross_pay – (5/100)*gross_pay)*

- Write default and parameterized constructors to initialize attributes of all classes.
- Write a class Person with attributes mentioned above, make it a parent of class Employee.
- Provide a default and parameterized constructor of Employee class, since employee is a child of Person, so make sure you use list initializer to pass values to the constructor of parent class.
- Make Employee class Abstract by declaring at least one pure virtual function calcSalary. You do not need to provide body for it as it is a pure virtual function and can only be implemented by child class of Employee. A pure virtual function is declared as below

```
virtual float calcSalary()=0;
```

- Write a class faculty, make it a child of Employee, declare its member variables

```
string designation;
string department;
string course;
```

- Implement calcSalary in Faculty class using the formula given above.
- Create a pointer of type Employee in main function. Since Employee is abstract class so we cannot instantiate it, but we can declare a pointer of type Employee.
- Create object of class Faculty by using pointer of Employee created in previous step. (use parameterized constructor of Faculty class.
- Calculate salary for the instance of the faculty class you created in the previous step.

Copy function headers from submission.cpp

Task# 3

Multilevel Inheritance

Write C++ class Drink. Publicly inherit "Drink" class to "Water" class and "Water" class to "Carbonated" class. i.e.

Water: Drink and **Carbonated: Water**

Class Drink should have the following attributes:

Flavor (string)
Temperature for best serve (float)
Price (float)
Expiry date (string)

- For **Drink** class, write default constructor to set all **string** values to “ “ and all float values to 0, and overloaded constructor for "Drink" to set Flavor, Temperature, price and Expiry date.
- Write getter/setter functions for Drink class.
- Inside **Water** class, declare a **string** variable **supplier**
- Write an overloaded constructor for Water class and a **Display** method to display all the attributes of Water.
- Inside **Carbonated** class, declare a **string** variable **type.**
- **Carbonated** class should have default, parameterized constructor and **void Display** function to display all the attributes of the class.