## Lab7: Classes

**Instructions**

* Data member names of each class should be the same as per mentioned in each question.
* Implement the required checks in member functions.
* Please read the questions carefully, read them twice even thrice to understand them completely. In case of any query, please raise your hands and we will be there to solve your query.
* Please concentrate, understand and code. Good Luck :)

# Task 1

Build a class **Sale** with private member variables double itemCost; // Cost of the item

double taxRate; // Sales tax rate

and functionality mentioned below:

1. Write a default constructor to set the member variable itemCost to 0 and taxRate to0.

## Sale( )

1. Write a parameterized constructor that accepts the parameter for each member variable such as **cost** for **ItemCost** and **rate** for **taxRate**

## Sale( double cost, double rate)

1. Generate only accessors for **itemCost** and d**taxRate**
2. Write a function **getTax( )** to calculate tax i.e take a product of itemCost and itemRate.

## double getTax( )

1. Write a function **getTotal( )** to calculate the total price of item i.e. take a sum of itemCost and getTax( ) (calling getTax() will return the calculated tax on item).

## double getTotal( )

**Task 2**

Declare a class **Box**. A box as you all know is something a cubical container. It has following attributes

* Length
* Width
* Height

In addition to these, a box can be made of different **materials** e.g. wood, card, metal etc. Further, more a box can have different **colors**. Declare them also as member variables of class.

* 1. Provide a default **Constructor**, a parameterized **Constructor**for the Box that takes all necessary values as arguments with the material as optional (if it is omitted the Box is considered to be made of Card – default value for the material.
  2. Provide getters for all attributes and setters for each too except for the material (material of box cannot be changed after when it has been created!!!).
  3. Provide a function **getVolume()**that calculates and returns the value of the volume of the Box.
  4. Also provide another function **getSurfaceArea(),**that calculates and returns the surface area of a Box.
  5. Provide a print function that Prints the following about the Box

Length:

Width:

Height:

Material:

Color:

Volume:

Surface Area:

* 1. Inside main, allocate a block of memory for 5 objects (using array). Read the necessary values from the user to populate array.
  2. Call functions **Print**() and Volume to display the data of boxes you just have saved in array.
  3. Read the index and the new height from the user, ask the user to provide index of the box to change its height by creating a function **update(Box b[], int size, int index, double height)**. Update the height of the Box present on the index provided by user, save the updated height right there, and print it again.

# Task 3

Build a class **CoffeeShots** (representing a cup of coffee) having the following attributes

* + type(string)
  + price(double)
  + volume(float)
  + size(char)
  1. Provide a parameterized constructor with arguments for type, price and volume. Provide a default value for the type parameter in constructor so that the user may omit providing this value when creating an instance. Initialize all data to the values provided in the argument list. However, since there is no argument for size, you will have to assign it a value by yourself. The size attribute will get a char value based on the following condition
     + The size is ‘s’ if the volume of the coffee is between 0 and 50ml
     + The size is ‘m’ if the volume is between 51 – 75ml
     + The size is ‘l’ if the volume is greater than 75ml
  2. Provide getters for each of these attributes, however, setter will be provided only for price.
  3. Build a method **upSize().** This method increases the volume of the coffee by 5ml and then resets the size accordingly so that the above conditions are still met. It also adds Rs. 5 to the price of the coffee.
  4. Provide another method **spillOver(float)** that takes the amount of coffee spilled (ml spilled) and then reduces the volume of coffee by that amount. For example if c is an instance of coffee and c.spillOver(3) is called, then it means that 3 ml of coffee is spilled and the volume gets reduced by this amount. Do not update the size or price.
  5. Write a non-member function (not belonging to this class) **create MyCofee().**This method prompts the user for all the details required to create the coffee instance and creates a dynamic instance of the coffee with the provided data. It then returns a pointer to this coffee instance.

Prototypes of the functions are mentioned in submission.cpp file, which are:

## Coffeeshots(double p,float v,string t=0) void setPrice(double price)

**double getPrice( ) float getVolume( ) string getType( ) char getSize( ) void upSize( )**

**float spillOver(float vol) void print( )**

**Coffeeshots& createMyCofee( )**

# Task 4

Write a class named as Car that manages the cars information .This class has following private data members:

* + - regNo: An String that represents the registration id of car.
    - entryTime: A integer that represents the entry time of the car in 24hformat.
    - exiteTime: A integer that represents the exit time of the car in 24hformat.

Generate getter setter for each data member of the class.

\*Note: Use validations: exit time of the car should always be greater than entry time of the car.

Write a class named as **ParkingGarage** that should be able to simulate a basic parking garage with the following functionality:

Parking Garage has parked cars. Your garage should have a fixed capacity (by default 5 cars). For each car stationed at your garage you will record its entry time (current day hour in 24h format), its registration number. A new car cannot be parked if the garage capacity is already full.

Whenever a new car arrives at your garage you will add it to the garage if the capacity is not full. When a car leaves the garage you will ask the user current time (current hour in 24 hour format) and charge him Rs. 20 per hour, and update noOfOccupied accordingly.

ParkingGarage contains following private data members:

* capacity:This is a constant data member of typei nteger.
* noOfOccupied: An integer that holds the value of occupied slots of parking.
* carPointer: A pointer of type Car that holds the informations of parked cars in garage.
* amountCollected : a double type variable which contains the total charges of all cars(calculated when cars are removed).
  1. Write a default constructor that initializes each data member of class such that capacity with 5, noOfOccupied with 0, amountCollected with 0 and carPointer with arrayoftype car having size of capacity.(Constant Members are initialized with Member Initialization list)

## ParkingGarage()

* 1. Write a parameterized Constructor that receives c capacity as argument and initializes each data member of class such that capacity with c , noOfOccupied with 0, amountCollected with 0 and carPointer with array of type car having size of c ParkingGarage(intc)
  2. Generate getter setter of each data member: such that noOfOccupied should never be greater thancapacity.
     + int getRemainingCapacity() const
     + double getAmountCollected()const
     + bool IsFull()const Should return whether Garage is full or not.
  3. Write a member function which add a new car to the Garage (having registration number in regnNumber and entry time in entryTime as arguments) if garage is not full. You should return true if car parked otherwise false.

bool ParkCar(const string &regnNumber,int entryTime ).