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 to 0.
   Sale( )
2. 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)

3. Generate only accessors for **itemCost** and **taxRate**
4. Write a function **getTax( )** to calculate tax i.e take a product of itemCost and itemRate.
   double getTax( )
5. 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.

2.1 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.

2.2 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!!!).

2.3 Provide a function **getVolume()** that calculates and returns the value of the volume of the Box.

2.4 Also provide another function **getSurfaceArea(),** that calculates and returns the surface area of a Box.

2.5 Provide a print function that Prints the following about the Box

Length:
Width:
Height:
Material:
Color:
Volume:
Surface Area:

2.6 Inside main, allocate a block of memory for 5 objects (using array). Read the necessary values from the user to populate array.

2.7 Call functions **Print**() and Volume to display the data of boxes you just have saved in array.

2.8 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

1. Define a class **Container** for storing integer values, regardless of the sequence and duplication checks. Your class should consist of the member variables.

   int * values; // A pointer to an int . This member points to the dynamically allocated array of integers (which you will be allocating in constructor).
   int capacity; // An integer that shows total capacity of your container
   int counter; // A counter variable which increments upon every insertion operation; shows total number of elements inserted yet in container

2. You would need a **parameterized** constructor with single parameter int c; i.e. it initializes the **capacity** variable, showing the total capacity of the container, and also allocating memory to array. Set the value of **counter** to 0.
   Container(int c)

Implement the functions mentioned below:

3. Function **bool isFull( ),** to check if the counter has reached to the max capacity of your array return true (counter = =capacity), else return false.

4. Function **void insert(int val)** takes single parameter, The function requires you to place that item in array, but before placing item, do call **isFull( )** to check if we have not reached to the capacity. Update the counter variable as well.
5. Function **bool search(int val)** takes 1 parameter. Provided value has to be searched in array, note that array is not passed in the parameter list of this function; because it is already accessible i.e. array is a part of this class. Return true if you found the value.
6. Function **void remove(int val)** removes the value after searching it in array. But the most important thing about this function is, that suppose user provided a value which is placed at 3rd index of array of size 5, so accomplishing the goal of removing value would require you to apply the logic of shifting the value from 4th index to 3rd and the value from 5th index to 4th. Don't forget to update counter.
7. Write a function **void Print( )** to print the values of array.

Task 4

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

- type (string)
- price (double)
- volume (float)
- size (char)

4.1 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 50 ml
- The size is 'm' if the volume is between 51 – 75 ml
- The size is 'l' if the volume is greater than 75 ml

4.2 Provide getters for each of these attributes, however, setter will be provided only for price.

4.3 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.

4.4 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.

4.5 Write a non-member function (not belonging to this class) **createMyCofee().** 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.

4.6 From main function, you just have to call createMyCofee( ), calls to other functions must be handled inside this function.

Prototypes of the functions 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 5

The absence of array bounds checking in C++ is a source of potential hazard. Write a class which will perform bounds checking on integer array.

Write a class IntegerList with private member variables as:

int *list; // A pointer to an int . This member points to the dynamically allocated array of integers (which you will be allocating in constructor).

int numElements; // An integer that holds the number of elements in the dynamically allocated array.

And public member functions

5.1 **IntegerList(int)** // The class constructor accepts an int argument that is the number of elements to allocate for the array. The array is allocated, and all elements are set to zero.

5.2 ~**IntegerList();** // Destructor to unallocated memory of list array.

5.3 **bool isValid(int);** // This function validates a subscript into the array. It accepts a subscript value as an argument and returns boolean true if the subscript is in the range 0 through numElements - 1. If the value is outside that range, boolean false is returned.

5.4 **void setElement(int, int**); // The setElement member function sets a specific element of the list array to a value. The first argument is the element subscript, and the second argument is the value to be stored in that element. The function uses isValid to validate the subscript. If subscript is valid, value is stored at that index, if an invalid subscript is passed to the function, the program aborts.

5.5 **int getElement(int);** // The getElement member function retrieves a value from a specific element in the list array. The argument is the subscript of the element whose value is to be retrieved. The function should use isValid function to validate the subscript. If the

subscript is valid, the value is returned. If the subscript is invalid, the program aborts. To abort, you can use exit(EXIT_FAILURE) function. (include library <cstdlib> for it)

Task 6

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 24h format.
- exiteTime: A integer that represents the exit time of the car in 24h format.

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 type integer.
- 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).

6.1 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 array of type car having size of capacity.(Constant Members are initialized with Member Initialization list)
ParkingGarage()

6.2 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(int c)

6.3 Generate getter setter of each data member: such that noOfOccupied should never be greater than capacity.

- int getRemainingCapacity() const
- double getAmountCollected() const

- bool IsFull()const Should return whether Garage is full or not.

6.4 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).

Task 7

Write a class named as Student to manages the Students information .This class has following private data members:

rollNo: An integer that represents the id of student.
name: A string that represents the name of the student.
address : A string that represents the address of the student.
batch: An integer that holds the batch number of student.

7.1 Write a default constructor that initializes each data member of class such that name with NULL, rollNo with 0, address with NULL and batch with 0.

Student()

7.2 Write a constructor that accepts the arguments for each data member such that int r assigned to rollNo, string n assigned to name, String ad assigned to address and int b assigned to batch.

Student(int r, const string &n, const string &ad, int b)

7.3 Generate getter setter of each data member : such that rollNo should not have negative value, name should never be greater than 50 characters, address should never be greater than 90 characters and batch should not have negative value.

bool setRollNo(int r)
int getRollNo()
bool setName(string &n)
string getName()
bool setAddress(string& ad)
string getAddress()
bool setBatch(int b)
int getBatch()

Write a class named as Section to manage a section of students. This class has following private data members:

sectionName: A character that represents the name of section.
studentPtr*: A pointer of type student pointing to the array of student objects created on heap memory.
sectionStrength: An constant integer that represents the maximum number of allowed students in a section.

currentStudent: an integer that represents the number of students currently present in a section.

7.4 Write a default constructor that initializes each data member of class such that sectionName with null character, studentPtr* with NULL and studentPtr* with 0 and currentStudent with 0

Section()

7.5 Write a parameterized constructor that that receives 's' strength and 'name' section Name as argu-ment,you need to initialize sectionStrength with s ,sectionName with name,currentStudent with zero, and studentPtr to memory having size 's' on heap of type Student.

Section (int s , char name)

7.6 Generate getter setter of each member variable: such that sId should not have negative value, sName should never be greater than 50 characters.

bool setSectionName(const string &n)
string getSectionName()
int getcurrentStudent()

In Section class write a member function for adding new Student, this function takes arguments: int r as rollNo, string n as name, string ad as address, int b as batch and adds• a student and also increments the data member currentStudent by one .

bool addStudent(int r, string &n, string &ad, int)

7.7 Write a member function which deletes the Student that was added at the last.

bool deleteStudent() ***Note: use getCurrentStudent()**

7.8 Write a member function which takes roll number as argument, searches a student and returns his name if student exist and returns "Not Found" if student does not exist.

string SearchStudent(int r) ***where r stands for roll number of the student**