## Lab7: Classes

# 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( )

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 **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

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

1. 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:

1. Function **bool isFull( ),** to check if the counter has reached to the max capacity of your array return true (counter = =capacity), else return false.
2. 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.
3. 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.
4. 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 3rdindex of array of size 5, so accomplishing the goal of removing value would require you to apply the logic of shifting the value from 4thindex to 3rdand the value from 5thindex to 4th. Don’t forget to update counter.
5. Write a function **void Print( )**to print the values of array.

**Task 3**

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 size;** the size of the dynamically allocated memory.

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

And public member functions

* 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.
  2. **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.
  3. **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.
  4. **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 beretrieved. 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.

# Task 4

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

**address**: A string that represents the address of the

**batch**: An integer that holds the batch number of student.

* 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 with0.

## Student()

* 1. 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 tobatch.

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

* 1. 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 negativevalue.

## 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.

* 1. 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 with0

## Section()

* 1. 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 oftype Student.

## Section (int s , char name)

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

## 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 )

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

## booldeleteStudent() \*Note: use getCurrentStudent()

* 1. 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.

## stringSearchStudent(int r) \*where r stands for roll number of the student