Chittagong University of Engineering and Technology
Dept. of Computer Science and Engineering

Lab Manual

Course Title: **Object Oriented Programming (Sessional)**
Course No: **Cse-144**
Credit Hour: **1.5**

Prepared By

Lamia Alam
Assistant Professor, Dept. of CSE, CUET

Omar Sharif
Lecturer, Dept. of CSE, CUET

General Laboratory Rules and Precautions for Computer Lab

The following general rules and precautions are to be observed at all times in the laboratory. These rules are for the benefit of the experimenter as well as those around him/her. Additional rules and precautions may apply to a particular laboratory.

- Use of PEN DRIVE, HARD-DISK DRIVE, OR EXTERNAL HARD DISK is strictly prohibited.
- FOOD, DRINK, OR USE OF TOBACCO IN ANY FORM is not allowed in the labs.
- TURN OFF CELL PHONES! If you need to use it, please take it to the hallway.
- No computer game is allowed.
- Lab users should maintain professional and courteous communication. Electronic devices should be used on a professional level. No obnoxious or belligerent behavior will be tolerated.
- Please operate the equipment with respect and care.
- Activities in the lab(s) that are considered by the lab aides to be abusive to the software, hardware, and or personnel may result in expulsion from the lab(s) and denial of future use of the lab(s).
- The lab aides are here to help when they can and to maintain the labs' operation. However, the lab aides are not here to do your work for you. The lab aides will refer students to their instructors at the lab aides discretion.
- Software may be installed by Computer Labs staff only. Do not install any software on your own. Files not put on by Computer Labs staff will be routinely removed.
- Do not modify any software or files. Do not overwrite the operating system, modify the autoexec.bat or config.sys or any other system parameters.
- For any hardware, software, printer, paper, or ink problems, please contact the aide.
- Any failure to follow these lab rules may result in the loss of your lab privileges.

Acknowledgement: Safety instructions adopted from Southwestern Oregon Community College.

Course Objectives:

- Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
- Design, implement, test, and debug simple programs in an object-oriented programming language.
- Describe how the class mechanism supports Inheritance, Polymorphism.

Course Outcomes:

- Develop program to illustrate basic concept of OOP features and C++ concept.
- Create and implement program using unary and binary operator overloading.
- Write program to implement concept of inheritance and polymorphism.
- Create program to implement concept of abstract class and virtual functions.
- Develop program using console I/O and file I/O.
- Develop and implement program using exception handling and templates.
- Apply all the programming concepts as and when required in the future application development of object oriented solutions for small systems

Course Activities:

- Lab Performance
- Quiz, Viva-Voce
- Course Project (Mini Project)

Requirements:

- Basic knowledge of any programming language (Preferred C or C++)
- Computer with proper configuration
- IDE for corresponding programming language

List of Experiments

SI No.	Experiment Name	Page No.
1	Introduction to Object-Oriented Programming and overview of C++ basics: Syntax, Data-types, I-O operations, Control structure, Loops	6-9
2	Overview of C++ basics: Arrays, Two dimensional array, Function, String	11-15
3	Familiarization with Standard Template Library (STL): Vector, Stack, Queue, Map, Set	17-21
4	Programs to understand the concepts of **Class, Object, Data member, Member functions**	23-26
5	Programs to implement the concepts of **Friend function, Friend class and Static members**	28-32
6	Review Lab (Performance Test)	34-36
7	Programs to implement the concepts of **Constructor, Copy Constructor and Destructor**	38-43
8	Programs to implement the concept of **Operator Overloading** by using member function and friend function	45-49
9	Programs to implement the concept of **Inheritance and Multiple inheritance**	51-55
10	Programs to implement the concept of **Polymorphism, Abstract Class, Virtual Function**	57-61
11		
12		
13	Quiz/ Viva / Project Presentation	

Project: The project is probably the most exciting part of this course and provides students with meaningful experience to design and implement a medium size system applying all the OOP techniques that we have learned throughout this course.

- Students will work in a group of two students for this project.
- The requirements will be announced and discussed in class.
- The project includes design, implementation, demonstration, and report writing.

Experiment:- 1

Practice Problems(Lab) for Lab-1

Exercise 1: Write a program to print your name where name is the input form the user.

```
// Cont.cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int ara[100], n, e=0, o=0;
    cin>>n;
    for(int i=0; i<n; i++){
        cin>>ara[i];
        if(ara[i]%2==0) e++;
        else o++;
    }
    cout<<"Even = "<<e<<" Odd = "<<o<<endl;
    return 0;
}
```

Exercise 2: Enter radius of a circle and find its diameter, circumference and area.

```
//Circle.cpp
#include<iostream>
using namespace std;
#define pi 3.1416
int main()
{
    double radius, diameter, cir, area;
    cin>>radius;
    diameter = 2*radius;
    cir = 2*pi*radius;
    area = pi*radius*radius;

    cout<<"Diameter = "<<diameter<<endl;
    cout<<"Circumference = "<<cir<<endl;
    cout<<"Area = "<<area<<endl;
    return 0;
}
```

Exercise 3: Write a program to find whether a number is even or odd.

```
//even_odd.cpp
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Please enter a number: ";
    cin>>n;
    if(n%2==0)cout<<n<<" is even"<<endl;
    else cout<<n<<"is odd"<<endl;
    return 0;
}
```

Exercise 4: Write a program to print all the prime numbers between 1 to n.

```
//prime.cpp
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    cout<<2<<' ';
    //unoptimized approach
    for(int i=3; i<=n; i++){
        int flag = 1;
         for(int j=2; j<=i/2;j++){
            if(i%j==0){
                flag = 0;
                break;
            }
         }
         if(flag == 1)cout<<i<<" ";
    }
    cout<<endl;
    return 0;
}
```

Exercise 5: Please run this code to see what this program prints.

```
//Star.cpp
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1; j<=n-i;j++){
            cout<<' ';
        }
        for(int j=1;j<=i;j++){
            cout<<'*';
        }
        cout<<endl;
    }
}
```

Practice Problems(Home) for Lab-1

These problems are designed to cover following C++ basics: Syntax, Data-types, I-O operations, Control structures, Loops

1. Write a C++ program to print your address and personal information.
2. Write a C++ program to enter marks of five subjects and calculate total, average and percentage
3. Write a C++ program to enter P, T, R and calculate Simple Interest.
4. Write a C++ program to find maximum between to three numbers.
5. Write a C++ program to input any character and check whether it is alphabet, digit or special character.
6. Write a C++ program to check whether the triangle is equilateral, isosceles or scalene triangle.
7. Write a C++ program to print all even number from 1 to n.
8. Write a C++ program to find sum of all prime numbers between range m to n.
9. Write a C++ program to print Fibonacci series up to n terms.

10. Print some star pattern.

```
*****     *      *****     *****     *****
*****     **      ****     ****     *****
*****     ***      ***     ***     *****
*****     ****      **     **     *****
*****     *****      *     *     *****
```

For better understanding of Loops and Nested Loops you can try to print following patterns. (Optional Exercises)

```
11111     01010     11111     11111     12345     12345     1
00000     01010     10001     11111     23456     23451     21
11111     01010     10001     11011     34567     34521     321
00000     01010     10001     11111     45678     45321     4321
11111     01010     11111     11111     56789     54321     54321
```

```
5 5 5 5 5 5 5 5 5     1          01 02 03 04 05     5          1
5 4 4 4 4 4 4 4 5     12         16 17 18 19 06     45         131
5 4 3 3 3 3 3 4 5     1234       15 24 25 20 07     345        13531
5 4 3 2 2 2 3 4 5     12345      14 23 22 21 08     2345       1357531
5 4 3 2 1 2 3 4 5     1234       13 12 11 10 09     12345      135797531
5 4 3 2 2 2 3 4 5     123
5 4 3 3 3 3 3 4 5     12
5 4 4 4 4 4 4 4 5     1
5 5 5 5 5 5 5 5 5
```

Sample Viva-Voce Questions

- What is Object Oriented Programming?
- What are basic features of OOP?
- What is the difference between object-oriented programming and procedural programming?
- Why C++ is known as object oriented programming? List any other languages which support OOP.
- Why namespaces are used in C++?
- Where are standard C libraries defined in C++?
- What is scope?
- What is a local variable?
- When should you use global variables?
- What is the role of scope resolution operator?
- What is the insertion operator and what does it do?
- What is the extraction operator and what does it do?
- What is the meaning behind the operator "<<" used in cout and the operator ">>" used in cin statements in C++?
- What is the difference between Assignment and Initialization?
- What are the two types of comments, and how do they differ?
- Can comments be nested?
- Can comments be longer than one line?

Experiment:- 2

Practice Problems(Lab) for Lab-2

Exercise 1: Write a program to count total number of even and odd elements in a array.

```cpp
// Even_odd.cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int ara[100], n, e=0, o=0;
    cin>>n;
    for(int i=0; i<n; i++){
        cin>>ara[i];
        if(ara[i]%2==0) e++;
        else o++;
    }
    cout<<"Even = "<<e<<" Odd = "<<o<<endl;
    return 0;
}
```

Exercise 2: Write a program to check whether two matrices are equal or not.

```cpp
//Matrices_check.cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int ara1[10][10], ara2[10][10], row, col;
    cin>>row>>col; //Take array size as input
    for(int i=0;i<row;i++){//iterating on row
        for(j=0;j<col;j++){//iterating on column
            cin>>ara1[i][j];//Taking input
        }
    }
    for(int i=0;i<row;i++){
        for(j=0;j<col;j++){
            cin>>ara2[i][j];
        }
    }
    int flag = 1; //Assuming that both arrays are equal
      for(int i=0;i<row;i++){
        for(j=0;j<col;j++){
            if(ara1[i][j]!=ara2[i][j]){//Checking elements
                    flag=0; // Assumption proved wrong
                    break;
            }
        }
    }
    if(flag==1)cout<<"Two matrices are equal"<<endl;
    else cout<<"Two matrices are not equal"<<endl;
}
```

Exercise 3: Write a program to find diameter, circumference and area of a circle using 3 different function.

```
//Circle.cpp
#include<bits/stdc++.h>
using namespace std;
#define pi 3.1416
double diameter(double r)//Function for calculating diameter
{
    return 2*r;
}
double circumference(double r)//Function for calculating circumference
{
    return 2*pi*r;
}
double area(double r);//This is function prototyping
int main()
{
    double radius;
    cin>>radius;
    cout<<diameter(radius)<<endl;
    cout<<circumference(radius)<<endl;
    cout<<area(radius)<<endl;
}
double area(double r)// Definition of prototyped function
{
    return pi*r*r;
}
```

Exercise 4: Write a program to find (i) length of string, (ii) copy one string to another, (iii) Concatenate two strings, (iv) Compare two strings.

```
//String.h
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string s1, s2, s3;
    cin>>s1>>s2;
    // Way of finding length of a string
    cout<<s1.size()<<" "<<s2.length()<<endl;
    // One string can be copied to another by simply using(=) operator
    s3 = s2;
    cout<<s3<<endl;
    // Concatenation can be done by simply using(+) operator.
    s3 = s1+" "+s2;
    cout<<s3<<endl;
    // Two string can be compared by using relational operators (==,
       !=, <, >, <= and >=)
    cout<<s1.compare(s2)<<endl;
}
```

Exercise 5: Write a program to check whether a string is palindrome or not.

```
//Palindrome.cpp
#include<bits/stdc++.h>
using namespace std;
void check_palindrome(string s)// Use defined function
{
    string s1 = s;
    // Built in function for reversing a string
    reverse(s1.begin(), s1.end());
    if(s1==s)cout<<s<<" is Palindrome"<<endl;
    else cout<<s<<" not Palindrome"<<endl;
}
int main()
{
    string s;
    cin>>s;
    int flag = 1;// Assuming string is palindrome
    int length = s.size();//Find length of the string
    for(int i=0;i<(length/2);i++){
        if(s[i]!=s[length-i-1]){// Checking first half with second half
           of the string
           flag = 0; //Assumption wrong
           break;
        }
    }
    if(flag==1)cout<<s<<" is Palindrome"<<endl;
    else cout<<s<<" not Palindrome"<<endl;
    check_palindrome(s);// Check palindrome using user defined function
}
```

*** Comment lines have been used to increase the readability of the code ***

Practice Problems(Home) for Lab-2

These problems are designed to cover following C++ basics:
Array, Two dimensional array, Function, String.

1. Write a program to find sum of all array elements.
2. Write a program to find maximum and minimum element in an array.
3. Write a program to find second largest element in an array.
4. Write a program to sort an array in (i) ascending order (ii) descending order.
5. Write a program to print all prime numbers between a given interval using functions.
6. Write a program to add (two/three) (integer/real) numbers using the concept of function overloading.
7. Write a program to convert (i) lowercase string to uppercase (ii) uppercase string to lowercase.
8. Write a program to find total number of alphabets, digits or special character in a string.
9. Write a program to count occurrences of a character in a string.
10. Write a program to find highest frequency character in a string.

Optional Exercises: We encourage you to solve these problems to clear your understanding on these topics.

1. Write a program to print all unique elements in an array.
2. Write a program to merge two array to third array.
3. Write a program to multiply two matrices.
4. Write a program to interchange diagonals of a matrix.
5. Write a program to find transpose of a matrix.
6. Build a simple calculator by using the concept of function.
7. Write a program to find frequency of each characters in a string.
8. Write a program to find reverse order of a string.
9. Write a program to remove all repeated characters from a given string.
10. Write a program that prints that you are ”Extra-Ordinary”.

Sample Viva-Voce Questions

- What do you mean by an Array?
- How to create an Array?
- What are the advantages and disadvantages of Array?
- Can we change the size of an array at run time?
- Can you declare an array without assigning the size of an array?
- What is the default value of Array?
- Can we declare array size as a negative number?
- When ArrayIndexOutOfBoundsException occurs?
- What is pointer?
- What is the difference between a reference and a pointer?
- What are the differences between the function prototype and the function definition?
- Do the names of parameters have to agree in the prototype, definition, and call to the function?
- If a function doesn't return a value, how do you declare the function?
- What are default arguments in functions in C++?
- What is function overloading?
- When you overload functions, in what ways must they differ?
- What is C-string?
- What is problem with C-string?
- Which header file is used to manipulate the string in C++?
- What can be used to input a string with blankspace?
- What does C++ append to the end of a string literal constant?
- What is the difference between unsigned int length() and unsigned int size()?
- How many parameters can a resize method take?
- Which method and operator do we use to append more than one character at a time?

Experiment:- 3

Practice Problems(Lab) for Lab-3

Familiarization with different Standard Template Library in C++.

Exercise 1: Familiarization with vector.

```cpp
//vector.cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n,x;
    vector<int>v;//initializing a vector
    v.push_back(10);//inserting an element into vector
    cin>>n;
    for(int i=1; i<=n; i++){// inserting n number of elements
        cin>>x;
        v.push_back(x);
    }
    for(int i=0; i<v.size(); i++){
        cout<<v[i]<<' ';
    }
    v.pop_back(); //remove elements from a vector from the back.
}
```

Exercise 1: Familiarization with queue.

```cpp
//queue.cpp
#include<bits/stdc++.h>
#include<queue>
using namespace std;

void print_element(queue<int> p)
{
    while(!p.empty()){ // Print elements of a queue
        cout<<p.front()<<' ';
        p.pop();
    }
    cout<<endl;
}
int main()
{
    int n, x;
    queue<int> myQueue;
    cin>>n;
    for(int i=1 ; i<=n; i++){
        cin>>x;
        myQueue.push(x); // insert element into a queue
    }
    print_element(myQueue);
    cout<<" Queue Size "<<myQueue.size()<<endl;
    cout<<"First element "<<myQueue.front()<<endl;
    cout<<"Last element "<<myQueue.back()<<endl;
    myQueue.pop();
    print_element(myQueue);
}
```

Exercise 3: Familiarization with stack.

```
//stack.cpp
#include<bits/stdc++.h>
#include<stack>
using namespace std;

void print_element(stack<int> p)
{
    while(!p.empty()){ // Print elements of a stack
        cout<<p.top()<<' ';
        p.pop();
    }
    cout<<endl;
}
int main()
{
    int n, x;
    stack<int> myStack;
    cin>>n;
    for(int i=1 ; i<=n; i++){
        cin>>x;
        myStack.push(x); // insert element into a stack
    }
    print_element(myStack);
    cout<<" Stack Size "<<myStack.size()<<endl;
    cout<<"First element "<<myStack.top()<<endl;
    myStack.pop();
    print_element(myStack);
}
```

Exercise 4: Familiarization with Map.

```
//Map.cpp
#include<bits/stdc++.h>
#include<map>
using namespace std;

int main()
{
    map<int, int>myMap;
    // insert elements in a map
    myMap.insert(pair<int, int>(1, 40));
    myMap.insert(pair<int, int>(2, 30));
    myMap.insert(pair<int, int>(3, 60));
    myMap[4]=20;
    myMap[7]=50;
   // Printing elements of map with key.
    map<int, int>::iterator itr;
    cout << "\tKEY\tELEMENT\n";
    for (itr = myMap.begin(); itr != myMap.end(); ++itr) {
        cout << '\t' << itr->first
             << '\t' << itr->second << '\n';
    }
    cout << endl;
}
```

Exercise: 5 Familiarization with set.

```
//Set.cpp
#include<bits/stdc++.h>
#include<set>
using namespace std;

int main()
{
    int n, x;
    set<int>mySet;
    cin>>n;
    for(int i=1; i<=n; i++){
        cin>>x;
        mySet.insert(x);
    }
    set<int>::iterator itr; //take a iterator to print element of a set
    for(itr=mySet.begin(); itr!=mySet.end(); ++itr)
    {
        cout<<*itr<<endl;
    }
}
```

*** Please go through the resource links to get further understanding of these standard libraries. ***

Practice Problems(Home) for Lab-3

These problems are designed to familiarize with Standard Template Library of C++: Vector, Stack, Queue, Set. Map

*** Please use the resource link go get better understanding and implement these home works. ***

1. Write a program to implement following functions in a vector. assign(), push_back(), pop_back(), insert(), erase(), swap(), clear() etc.
[Resource Link (Vector)]

2. Write a program to implement following functions in a queue. size(), empty(), push(), back(), front() etc.
[Resource Link (Queue)]

3. Write a program to find to implement following functions in a stack. top(), empty(), push(), swap() etc.
[Resource Link (Stack)]

4. Write a program to find to implement following functions in a set. insert(), begin(), end(), lower_bound(), upper_bound(), size() etc.
[Resource Link (Set)]

5. Write a program to find to implement following functions in a map. insert(), begin(), end(), erase(), size() etc.
[Resource Link (Map)]

Optional Exercises: We encourage you to learn following standard template libraries. This will be really helpful for your programming carrier.

- priority_queue
- list
- multi_map
- unordered_map
- set
- multi_set
- pair

[Resource Link (STL)]

Sample Viva-Voce Questions

- What is STL?
- Enlist the Components of Standard Template Library.
- What is a Vector in STL?
- How can we declare an empty vector?
- How to remove elements from a vector from the back?
- What is Queue in STL?
- Find the size of a queue.
- Print front element and last element of queue.
- Write the piece of code to empty a queue.
- What is Stack?
- Write a piece of code to print all the elements of a stack.
- What is map and why it is necessary ?
- Is duplicate data allowed in set?
- From which STL we can insert/remove data from anywhere?
- What are Iterators in C++ STL?
- What are Lists in C++ STL?
- Which of the STL containers store the elements contiguously (in adjacent memory locations)?

*** Fell free to read more about Standard Template Library of C++. STL makes C++ more suitable for competitive programming than other languages. ***

Experiment:- 4

Practice Problems(Lab) for Lab-4

Things we will try to learn in this lab:

- What is class? How to define a class.
- Create data member and member function (Method) of a class.
- Visibility of data member and member function(Public and Private access)
- By creating object access members of a class.

Exercise 1: Write a C++ program to define a class **BOX** with,

Data member: [**length, breadth and height**]
Member function: [**input_value(), print_value() and volume()**].

Now find the volume of a box by accessing the members of this class using its object.

```
//Class_object.cpp
#include<bits/stdc++.h>
using namespace std;

class BOX
{
    public: // public access of the members
      double length, breadth, height;// data members

      // defining member functions
      void input_value()
      {
          cout<<"Enter three sides of a box: "<<endl;
          cin>>length>>breadth>>height;
      }
      void print_value()
      {
          cout<<"Length : "<<length<<endl;
          cout<<"Breadth : "<<breadth<<endl;
          cout<<"Height : "<< height<<endl;
      }
      double volume()
      {
          double v=length*breadth*height;
          return v;
      }
};
int main()
{
    BOX myBox; //creating a object
    BOX ara[100]; // creating an array of objects
    // Accessing members of class through its object
    myBox.input_value(); //taking the inputs

    myBox.print_value();//printing the values

    double vol= myBox.volume(); //calculating volume
    cout<<"Volume of the box: "<<vol<<endl;
}
```

Exercise 2: Write a C++ program to understand public and private access of class data members.

```
//public_private.cpp
#include<bits/stdc++.h>
using namespace std;

class myTest
{
    private:
       int a,b,c;
    public:
        void access_private()
        {
            cin>>a>>b>>c;
            cout<<a<<' '<<b<<' '<<c<<endl;
        }
};
int main()
{
    myTest v;
    cin>>v.a>>v.b>>v.c;
    // This will give us an error because we can not access the private
        data members outside of a class. To access this we must have a
        public member function.

    v.access_private();//Public function to access private members

}
```

*** For better understanding please read your reference text book. Fell free to search on internet because it is the best source of learning. ***

Practice Problems(Home) for Lab-4

These problems are designed to clear your understanding on Class and Object.

1. Write a class having two private variables and one member function which will return the area of the rectangle.

2. Write a C++ Program to define a class batsman with the following specifications:
Private members:

batsman_code: 4 digits code number
batsman_name: 20 characters(string)
total_innings, notout_innings, toal_runs: integer type
batting_avg: [toal_runs/(total_innings-notout_innings)] (formula to calculate batting average)
calcavg(): Function to compute batavg

Public members:

readdata(): Function to accept value from batsman_code, batsman_name, total_innings, notout_innings, total_runs and invoke the function calcavg().

displaydata(): Function to display the data members on the screen.

Access all the data members and member functions to calculate batting average of a batsman by creating its object.

[Resource Link 1]
[Resource Link 2]
[Resource Link 3]

Sample Viva-Voce Questions

- What is a class?
- What is an object?
- What is an inline function?
- What are/is the operator/operators used to access the class members?
- Which access specifier/s can help to achieve data hiding in C++?
- When a class member is defined outside the class, which operator can be used to associate the function definition to a particular class?
- What are access modifiers?
- What is the difference between structure and a class?
- Write down two main difference between structure and a class.
- What is the default access modifier in a class?
- Where a class member function can be defined?
- Can we declare a member function private?
- What is the difference between public and private data members?
- Do class declarations end with a semicolon? Do class method definitions?
- Write a code to show different way of declaring a member function.

Experiment:- 5

Practice Problems(Lab) for Lab-5

Things we will try to learn in this lab:

- What is friend function and friend class? How to define them.
- Create friend function and friend class and by using them access data member of a class.
- Familiarization with static data member and static member function.

Exercise 1: Write a class **Rectangle** having two private variables **(length, width)**. Now calculate the area of a triangle by using,

(i) [**Friend Function**]
(ii) [**Friend Class**].

(i) Friend Function

```
// Friend_function.cpp
#include<bits/stdc++.h>
using namespace std;

class Rectangle
{
    double length, height;
public:
    void take_input()
    {
        cout<<"Enter length and height : ";
        cin>>length>>height;
    }
    friend void area(Rectangle r); //Declaration of friend function
};
//Accessing private members using friend function
void area(Rectangle r)
{
    cout<<"Area: "<<r.length*r.height<<endl;
}
int main()
{
    Rectangle a;
    a.take_input();
    area(a);
}
// Sample Input Output

//Enter length and height : 3.1 4.2
// Area: 13.02
```

(ii) Friend Class

```
//Friend_class.cpp
#include<bits/stdc++.h>
using namespace std;

class Rectangle
{
    double length, height;
public:
    void take_input()
    {
        cout<<"Enter length and height : ";
        cin>>length>>height;
    }
    friend class Rectangle_friend;
};

class Rectangle_friend
{
public:
    void display(Rectangle r)
    {
        cout<<"Displaying by using a friend class: ";
        cout<<r.height<<" "<<r.length<<endl;
    }
    void area(Rectangle r)
    {
         cout<<"Area: "<<r.length*r.height<<endl;
    }
};

int main()
{
    Rectangle a;
    a.take_input();
    Rectangle_friend b;
    b.display(a);
    b.area(a);// Calculating area using friend class
}
```

Exercise 2: Familiarization with **Static Members** of a class.

```
//Static_member.cpp
#include<bits/stdc++.h>
using namespace std;

class shared
{
public:
    static int a;
    int b;
    void set_value(int i, int j)// Value initialization using
        constructor
    {
        a=i;
        b=j;
    }
    void display()
    {
        cout<<"a: "<<a<<" b: "<<b<<endl;
    }
    static void display2()
    {
        // Static member function can only access static data members
        a=4;
        cout<<a*2<<endl;
    }
};

int main()
{
    shared x, y ;
    x.set_value(1,1);
    x.display();
    y.set_value(2,2);
    y.display();
    x.display();
    //Static member function can be accessed without using object
    shared::display2();
    // Just look at the value of a and b you will understand what
        static means.
}
// a: 1 b:1
// a: 2 b:2
// a: 2 b:1
//8
```

Practice Problems(Home) for Lab-5

These problems are designed to clear your understanding on Friend function, Friend class and Static data members.

1. Write a program to check whether a number is even or odd by using the concept of friend function and friend class.

2. Write a program to add two complex number by using friend function and friend class.

3. Write a program to define a class **Student** with the following specifications,

roll_number: 6 digits roll number
static member count: To keep track on number of object
static function getcount(): return the value of count
function getnumber(): To take roll number as input
function putnumber(): To show the roll number

Access all the data members and member functions using the objects of class Student.

Sample Input/Output

```
Initially number of objects: 0
Enter number of entry: 2
Enter Roll No.: 1804001
Enter Roll No.: 1804002

Finally number of objects: 2
Roll No.: 1804001
Roll No.: 1804002
```

[Resource Link 1]
[Resource Link 2]

Sample Viva-Voce Questions

- What is a friend function?
- What is a friend class?
- Write a code to show how to define a friend function/class.
- When one should use friend function and friend class.
- Does structure support friend function/class.
- If Dog is a friend of Boy, is Boy a friend of Dog?
- If Dog is a friend of Boy, and Terrier derives from Dog, is Terrier a friend of Boy?
- If Dog is a friend of Boy and Boy is a friend of House, is Dog a friend of House?
- Where must the declaration of a friend function appear?
- What is role of static keyword on class member variable?
- Explain the static member function.
- Will you be able to access a non static data member by a static member function?
- What are differences between a normal member function and a static member function.
- Whether non static method can use static members?
- Can static member variables be private?

Experiment:- 6

Review Lab (Performance Test)

Things we will try to learn in this lab:

- Review the lessons that we have learnt in our previous labs.
- Performance test on previous concept.

Problem A: Sum of Even/Odd

Description:

Write an object oriented program that reads a set of non-negative integer numbers and output the sum of odd numbers and even numbers from that. To create the program, you have to write a class name **IntegerFactory** where there will be an integer array as private variable. There will also be public member functions that will responsible for providing the sums.

Input Format:

Program will read an integer n, followed by n integers and an integer c for choice to show either the sum of even number or the sum of odd numbers of these n integers. c=0 for sum of odd numbers and c=1 for sum of even numbers.

Constraints:

$0 < n <= 10000, 0 <= c <= 1$

Sample Input Output:

Sample Input	Sample Output

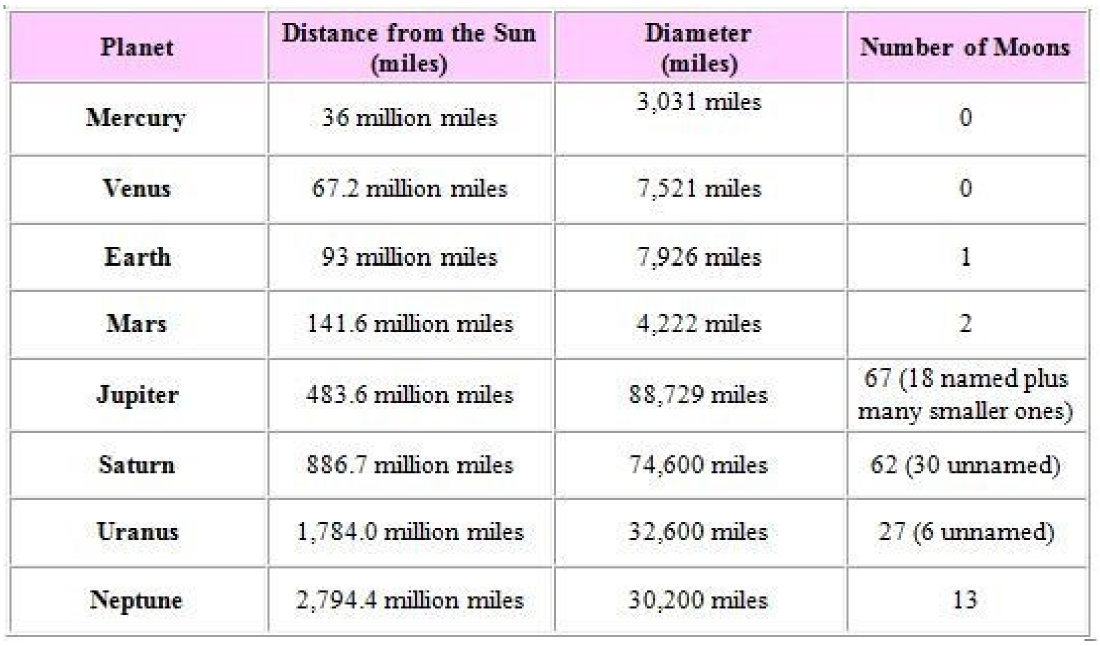
5
1 2 3 4 5
0
	9

8
2 1 3 6 2 10 5 2
1
	22

Problem B: Planet

Description:

Our solar system consists of the sun, eight planets, moons, many dwarf planets (or plutoids), an asteroid belt, comets, meteors, and others. The sun is the center of our solar system; the planets, their moons, a belt of asteroids, comets, and other rocks and gas orbit the sun. The eight planets that orbit the sun are (in order from the sun): Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune. Some useful details about planets are given below,



Planet	Distance from the Sun (miles)	Diameter (miles)	Number of Moons
Mercury	36 million miles	3,031 miles	0
Venus	67.2 million miles	7,521 miles	0
Earth	93 million miles	7,926 miles	1
Mars	141.6 million miles	4,222 miles	2
Jupiter	483.6 million miles	88,729 miles	67 (18 named plus many smaller ones)
Saturn	886.7 million miles	74,600 miles	62 (30 unnamed)
Uranus	1,784.0 million miles	32,600 miles	27 (6 unnamed)
Neptune	2,794.4 million miles	30,200 miles	13

You have to create a class **Planet** which has following data members and member functions.

Private Members:

- planer_name: string
- distance (from sun): double
- diameter: double
- number_of_moons: int

Create setter and getter functions for each element,

Member Functions:

- get_planet_name(), set_planet_name()
- get_no_of_moons(),set_no_of_moons()
- get_diameter(),set_diameter()
- get_distance_from_sun(),set_distanc_from_sun()

You also have to create a friend function called **light()** that takes as an argument an object of type Planet and returns the number of seconds that it takes light from sun to reach the planet.
(Assumes that light travels the sun travels at 1, 86, 000 miles per second and that distance from sun is specified in miles.)

Input Format:

First line will contain the number of T test cases. Each test case will contain four inputs describing the **planet name(P), distance from sun(DS), diameter(D) and number of moons(M)**.

Constraints:

$0 < T <= 100, 0 <= DS <= 1000, 0 < D < 10^5, 0 < M < 67$

Sample Input Output:

Sample Input	Sample Output
2	
Earth 93 7926 1	500
Mercury 36 3031 0	193.548

Experiment:- 7

Practice Problems(Lab) for Lab-7

Things we will try to learn in this lab:

- What is constructor and destructor? How to define them.
- Learn about default constructor, constructor overloading, copy constructor and implement those ideas.
- Understand the concept of constructor and destructor.

Exercise 1: Write a C++ program to add two complex number using constructor.**(Apply the concept of constructor overloading)**

```
//Adding two complex number
#include<bits/stdc++.h>
using namespace std;

class Complex
{
    float real, img;
public:
    Complex(){};//Default constructor

    //Constructor overloading
    Complex(float x) //One parameter
    {
        real=img=x;
    }
    Complex(float x, float y) // Two parameter
    {
        real=x;
        img=y;
    }
    Complex(Complex &c) //Copy constructor
    {
        real=c.real;
        img=c.img;
    }
    friend void sum(Complex, Complex);
};
void sum(Complex c1, Complex c2)
{
    Complex c3;
    c3.real = c1.real + c2.real;
    c3.img = c1.img + c2.img;
    cout<<"Sum of two complex number is: "<<c3.real<<" + "<<c3.img<<"i";
}
```

```
int main()
{
    Complex c1; //Calling default constructor
    Complex c2(3); // Call constructor with one parameter
    Complex c3(10, 15); // Call constructor with two parameter
    sum(c2, c3);
}

// Sum of two complex number is = 13 + 15i
```

Exercise 2: Define a class **Play** with the following specifications.
Private Member:
Playcode: integer
PlayTitle: string
Duration: float
Noofscenes: integer
Public member functions:

- A constructor function to initialise Duration as 45 and Noofscence as 5.
- Newplay() unction to accept values for Playcode and PlayTitle.
- Newinfo() function to assign the values of Duration and Noofscenes with the help of corresponding values passed as parameters to this function.
- Showplay() function to display all the data member on the screen.

```
//Play.cpp
#include<bits/stdc++.h>
using namespace std;

class Play
{
    int Playcode, No_of_scenes;
    string PlayTitle;
    float Duration;
public:
    Play()
    {
        Duration = 45;
        No_of_scenes = 5;
    }
    void Newplay()
    {
        cout<<"Enter value of Playcode and PlayTitle:\n";
        cin>>Playcode>>PlayTitle;
    }
    void Newinfo(int d, int n)
    {
        Duration=d;
        No_of_scenes =n;
    }
    void Showplay()
    {
        cout<<"Displaying all data members:\n";
        cout<<"Play Code: "<<Playcode<<endl;
        cout<<"Play Title: "<<PlayTitle<<endl;
        cout<<"Duration: "<<Duration<<endl;
        cout<<"No. of Scenes: "<<No_of_scenes<<endl;
    }
};
int main()
{
    Play P;
    P.Newplay();
    P.Newinfo(20,2);
    P.Showplay();
}

// Play Code: 10
// Play Title: Football
// Duration: 20
// No. of scenes: 2
```

Practice Problems(Home) for Lab-7

These problems are designed to clear your understanding on constructor and destructor.

Problem:1

A University wishes to keep information on its students. The proposed **Student class** has the following instance variables:

studentNo: String
studentName: String
dateOfBirth: String
tariffPoints: Integer

Tariff Points represents the entry qualification achieved by a student, which is a number between 20 and 280.
A class variable is also required, called **noOfStudents**, which will be incremented each time a Student instance is created. Write program to perform the following,

1. Show the declaration of the Student class, including any setter and getter methods.
2. Declare two constructors as follows; both constructors should increment the class variable appropriately:
   - The first is a default constructor that has no parameters and sets the instance variables to either "not known" for the strings, 20 for the integer and 1st January 1995 for the date.
   - The second takes 4 parameters, one for each of the instance variables.
3. Show how both constructors could be used to instantiate an object.

Sample Input Output

Press

1. for creating student instance with unknown value.
2. for creating student instance with known value
3. to display.
4. to exit.

Enter Your Choice: 1
Press

1. for creating student instance with unknown value.
2. for creating student instance with known value
3. to display.

```
   4. to exit.

Enter Your Choice: 2
Enter Student Details:
1          Student No: 1404122
2          Name: Rahim
3          Date of Birth: 2nd February 1995
4          Tariff Points: 210

Press

   1. for creating student instance with unknown value.

   2. for creating student instance with known value

   3. to display.

   4. to exit.

Displaying Students Details:

Student No          Name          Date of Birth          Tariff Points
Not Known           Not Known     1st January 1995        20
1404122             Rahim         2nd February 1995        210

Number of Instance Created: 2
```

Problem:2

A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the required copies is displayed, otherwise the message Sorry! These many copies are not in stock is displayed.
Design a system using a class called stock with suitable member functions and constructors.

Sample Viva-Voce Questions

- What is a constructor?
- What is default constructor?
- What are the characteristics of constructors?
- List some of the special properties of the constructor functions.
- Explain the role of a default constructor? When is it considered equivalent to a parameterized constructor?
- What is parameterized constructor?
- Is it possible to overload constructor?
- What is a copy constructor? What is it's significance?
- Describe the importance of destructor. List some of the special properties of destructor
- Does destructor take any argument?
- Is it possible to overload destructor?
- What do you think is the advantage of declaring the constructor and destructor functions for public member access?
- Differentiate between Constructor and Destructor function with respect to Object Oriented Programming.
- When a compiler can automatically generate a constructor if it is not defined then why is it considered that writing constructor for a class is a good practice?

Experiment:- 8

Practice Problems(Lab) for Lab-8

Things we will try to learn in this lab:

- Why operator overloading in important in oop? How to define it.
- Overload unary and binary operators.
- Understand and implement the concept of operator overloading by using member function and friend function.

Exercise 1: Write a C++ program to subtract complex number using operator overloading and the concept of member function.

```
//Subtract two complex number
#include<bits/stdc++.h>
using namespace std;

class cmplx
{
    double real, img;
public:
    cmplx(){} // Default constructor
    cmplx(double x, double y)
    {
        real = x;
        img = y;
    }
    cmplx operator- ( cmplx b)// overload operator as member function
    {
        cmplx temp;
        temp.real = real - b.real;
        temp.img = img - b.img;
        return temp;
    }
     void show()
    {
        cout<<real<<img<<"i"<<endl;
    }
};
int main()
{
    int r1, i1, r2, i2;
    cin>>r1>>i1>>r2>>i2;
    cmplx c1(r1, i1), c2(r2, i2), c3;
    c3 = c1 - c2;
    c3.show();

}
// 10, 1, 11, 2
//  1-1i
```

Exercise 2: Write a C++ program to Overloaded **++operator** in both prefix and postfix by using,

a) Member function
b) Friend function

(a) Using member function

```
#include<bits/stdc++.h>
using namespace std;

class PrePost
{
    int a;
public:
    PrePost()
    {
        a = 0;
    }
    void show()
    {
        cout<<a<<endl;
    }
    PrePost operator++()
    {
        PrePost z;
        cout<<"Prefix "<<endl;
        z.a= ++a;
        return z;
    }
    PrePost operator++(int)
    {
        PrePost z;
        cout<<"Postfix"<<endl;
        z.a = a++;
        return z;
    }
};
int main()
{
    PrePost p,x;
    x.show(); p.show();
    x=++p;
    x.show(); p.show();
    x=p++;
    x.show(); p.show();
}
//x=0  p=0
//x=1  p=1
//x=1  x=2
```

(a) Using friend function

```
#include<bits/stdc++.h>
using namespace std;

class PrePost
{
    int a;
public:
    PrePost()
    {
        a = 0;
    }
    void show()
    {
        cout<<a<<endl;
    }
    friend PrePost operator++(PrePost &x);
    friend PrePost operator++(PrePost &x, int);
};
PrePost operator++(PrePost &x)
{
    PrePost z;
    cout<<"Prefix "<<endl;
    z.a= ++x.a;
    return z;
}
PrePost operator++(PrePost &x, int v)
{
    PrePost z;
    cout<<"Postfix "<<endl;
    z.a = x.a++;
    return z;
}
int main()
{
    PrePost p,x;
    x.show(); p.show();
    x=++p;
    x.show(); p.show();
    x=p++;
    x.show(); p.show();
}
//x=0 p=0
//x=1 p=1
//x=1 x=2
```

Practice Problems(Home) for Lab-8

These problems are designed to clear your understanding on operator overloading.

Problem:1
Define a class **Distance** with feet and inch and with a print function to print the distance.

a) overload < **operator** compares two distances using member function.
b) overload + **operator** to add two Distances using friend function.

Problem:2
You can do this for fun, only fun.
Write a C++ program to Overloaded + **operator** to subtract two complex number.

*** Follow the slide provided in the theory class. It covers most of the varieties of operator overloading such as,

- overload unary operators
- overload binary operators
- overloading using friend functions and member functions
- overload input and output operators
- overload relational operators

*** This topics are important for final lab test and quiz-viva ***

Sample Viva-Voce Questions

- What is operator overloading?
- Which keyword can be used for overloading?
- What are all the operators that cannot be overloaded?
- What are the restrictions on operator overloading.
- Which of the operators are overloaded by default by the compiler in every user defined classes even if user has not written?
- Which of the operators should be preferred to overload as a global function rather than a member method?
- How does C++ compiler differs between overloaded postfix and prefix operators?
- Which of the operator functions cannot be friend, i.e., must be a member function?
- What are the benefits of operator overloading?
- What operators can/cannot be overloaded?
- Can I create a operator** for "to-the-power-of" operations?
- What are some guidelines / rules of thumb for overloading operators?
- Differentiate between operating overloading using friend function and member function.

Experiment:- 9

Practice Problems(Lab) for Lab-9

Things we will try to learn in this lab:

- What is inheritance? Why use inheritance?
- Usages of access identifiers in terms of inheritance.
- Understand and implement the concept inheritance, multiple inheritance and solve different types of problems related to inheritance.

Exercise 1: Create two classes named Mammals and MarineAnimals. Create another class named BlueWhale which inherits both the above classes. Now, create a function in each of these classes which prints "I am mammal", "I am a marine animal" and "I belong to both the categories: Mammals as well as Marine Animals" respectively. Now, create an object for each of the above class and try calling,

- function of Mammals by the object of Mammal
- function of MarineAnimal by the object of MarineAnimal
- function of BlueWhale by the object of BlueWhale
- function of each of its parent by the object of BlueWhale

```cpp
//Inheritance1.cpp
#include<bits/stdc++.h>
using namespace std;

class Mammals
{
public:
    void mammal_print()
    {
        cout<<"I am a Mammal"<<endl;
    }
};
class Marine_animals
{
public:
    void marine_print()
    {
        cout<<"I am a Marine animal"<<endl;
    }
};
class BlueWhale: public Mammals, public Marine_animals
{
public:
    void BlueWhale_print()
    {
        cout<<"I belong to both the categories: ";
        cout<<" Mammals as well as Marine Animals"<<endl;
    }
};
int main()
{
    Mammals m;
    Marine_animals ma;
    BlueWhale b;
    cout<<"Accessing the object of own class"<<endl;
    m.mammal_print(); ma.marine_print(); b.BlueWhale_print();

    cout<<endl<<endl<<"Accessing the object of Base class"<<endl;
    b.mammal_print(); b.marine_print();
}
```

Exercise 2: Write a C++ program to do the following,

1. Student is a base class, having two data members: StudenId and Name; Student Id is an integer and name is a string.
2. Science and Arts are two derived classes, having respectively data items marks for Physics, Chemistry, Mathematics and marks for English, History, Economics.
3. Read appropriate data from the screen for a science and a arts students.
4. Display Student Id, name, marks for science student first and then for arts student.

```
// Student.cpp
#include<bits/stdc++.h>
using namespace std;
class student
{
protected:
    int StudentId;
    string name;
public:
    void getdata()
    {
        cout<<"Enter Student Id and Name "<<endl;;
        cin>>StudentId;
        getchar();
        getline(cin, name);
    }
    void show()
    {
        cout<<"Student Id: "<<StudentId<<endl;
        cout<<"Name "<<name<<endl<<endl;
    }
```

```cpp
};
class science: public student
{
    int phy, chem, math;
public:
    void getdata()
    {
        cout<<"Enter the marks of Physics, Chemistry and
           Mathematics"<<endl;
        cin>>phy>>chem>>math;
    }
    void show()
    {
        cout<<"Physics mark: "<<phy<<endl;
        cout<<"Chemistry mark: "<<chem<<endl;
        cout<<"Mathematics mark: "<<math<<endl<<endl;
    }
};
class arts: public student
{
    int eng, his, eco;
public:
    void getdata()
    {
        student::getdata();// calling the base class function
        cout<<"Enter the marks of English, History and Economics"<<endl;
        cin>>eng>>his>>eco;
    }
    void show()
    {
        student::show();
        cout<<"English mark: "<<eng<<endl;
        cout<<"History mark: "<<his<<endl;
        cout<<"Economics mark: "<<eco<<endl;
    }

};
int main()
{
    science s;
   /* Important concept: As both base and derived class have same
      function name we use a method to resolve this situation. */
    s.student::getdata();
    s.getdata();
    s.student::show();
    s.show();
    arts a;
    a.getdata();
    a.show();
}
```

Practice Problems(Home) for Lab-9

These problems are designed to clear your understanding on inheritance and access identifiers

Problem:1

Define a class named **Fruit** with a data member to calculate the number of fruits in a basket. Create two other class named **Apples** and **Mangoes** to calculate the number of apples and mangoes in the basket. Print the number of fruits of each type and the total number of fruits in the basket using the concept of inheritance.

Problem:2

We want to calculate the total marks of each student of a class in Physics,Chemistry and Mathematics and the average marks of the class.The number of students in the class are entered by the user. Create a class named **Marks** with data members for roll number, name and marks. Create three other classes inheriting the Marks class, namely Physics, Chemistry and Mathematics, which are used to define marks in individual subject of each student.

*** Follow the slide provided in the theory class. It covers other topics of inheritance such as,

- hierarchical inheritance
- what is diamond problem and ways to resolve this problem
- insights on access identifiers
- and some topics of book as well.

*** This topics are important for final lab test and quiz-viva. It is good practice to read the text book. Read the text book as well please. ***

Sample Viva-Voce Questions

- What is inheritance?
- What is Base class / super class/ parent class?
- What is Subclass/ derived class/ child class?
- Write general notation of inheritance in C++.
- Explain the concept of Inheritance vs Access with example.
- How many Access specifier are there in C++?
- What are the advantages of inheritance ?
- What are the types of inheritance ?
- Is inheritance possible in C ?
- What is the syntax of inheritance of class?
- What is private inheritance?
- What is protected inheritance?
- What is single level inheritance?
- What is multilevel inheritance?
- What is hierarchical inheritance?
- Which type of inheritance leads to diamond problem?
- What are the ways to resolve the diamond problems?
- If a derived class object is created, which constructor is called first?
- Discuss the order of constructor and destructor call in base and derived class.

Experiment:- 10

Practice Problems(Lab) for Lab-10

Things we will try to learn in this lab:

- What is polymorphism?
- Learn about virtual function, pure virtual function and abstract class.
- Understand and implement the concept polymorphism and solve different types of problems related to virtual function and abstract class.

Exercise 1: Create a class named **Shape** with a function that prints “This is a shape”. Create two other classes named Rectangle and Triangle which inherit the Shape class and have the same function that prints “This is rectangle” and “This is triangle” respectively. Now, write a program that access the functions of derived class by using the object of base class. **(Use the concept of virtual function)**

```
//virtual_function.cpp

#include<bits/stdc++.h>
using namespace std;

class shape
{
public:
    virtual void print()
    {
        cout<<"This is shape"<<endl;
    }
};

class rectangle : public shape
{
public:
    virtual void print()
    {
        cout<<"This is rectangle"<<endl;
    }
};

class triangle: public shape
{
public:
    virtual void print()
    {
        cout<<"This is triangle"<<endl;
    }
};

int main()
```

```
{
    shape *s, ss;
    rectangle r;
    triangle t;

    s = &ss; // Point to the base class
    s->print();
    s = &r; // Point to the derived class rectangle
    s->print();
    s = &t;// Point to the derive class triangle
    s->print();
}
// This is Shape
// This is Rectangle
// This is Triangle
```

Exercise 2: Create an abstract base class shape with two members base and height, a member function for initialization and a pure virtual function to compute area(). Derive two specific classes Triangle and Rectangle which override the function area(). Use these classes in a main function and display the area of a triangle and a rectangle.

```
// abstract_class.cpp
#include<bits/stdc++.h>
using namespace std;

class shape
{
protected:
    double base , height;
public:
    void get_data()
    {
        cout<<"Enter base and height: "<<endl;
        cin>>base>>height;
    }
    virtual void area()=0;
};
class rectangle: public shape
{
public:
    void area()
    {
        cout<<"Area of rectangle is: ";
        cout<<base*height<<endl;
    }
};

class triangle: public shape
```

```
{
public:
    void area()
    {
        cout<<"Area of triangle is: ";
        cout<<0.5*base*height<<endl;
    }
};

int main()
{
    rectangle r;
    //Calculate area of the rectangle
    triangle t;
    r.get_data();
    r.area();
    // Calculate area of the triangle
    t.get_data();
    t.area();
}
```

Practice Problems(Home) for Lab-10

These problems are designed to clear your understanding on inheritance and access identifiers

Problem:1

Suppose we need to model two-dimensional shape objects, such as rectangles, squares, triangles, ellipses, and circles. Some of these shapes are relateda square is a special kind of rectangle, and a circle is a special kind of ellipse. Each kind of shape has unique properties that distinguish it from the others; for example, a rectangle has a length and width, but just one value specifies the size of a square. A circle is a kind of ellipse, and a square is a kind of rectangle, Circles, ellipses, squares, rectangles, and triangles are all kinds of shapes. Here Rectangle and Ellipse are concrete classes. We can have an actual rectangle. Shape is the only abstract class in our shape hierarchy. Each shape object can compute its area. We will maintain a collection of shape objects to compute the total area of all the shapes in our collection from main().

*** Read the problem description and solve it my students. ***

Sample Viva-Voce Questions

- What is Polymorphism?
- How compile-time and run-time polymorphism can be achieved?
- Explain types of Polymorphism.
- What is method overloading?
- What is method overriding?
- Differentiate between overloading and overriding.
- Define virtual function
- Is virtual functions are hierarchical?
- What is a pure virtual function?
- When a virtual function is declared as pure then all the derived classes must override it, is it true?
- Write down the syntax of pure virtual function.
- What is abstract class?
- Is it possible to create object of an abstract class?
- Can we have a constructor as Virtual?