

Experiment 1

a Write a C++ program to declare class student & accept & display the data for one object

```
#include<iostream>
using namespace std;
class student()
{
    string name, clg;
    int roll-no;
public:
    void accept()
    {
        cout<<"enter name : ";
        getline(cin, name);
        cout<<"enter roll no. ";
        cin>>roll_no;
        cout<<"enter class : ";
        getline(cin, clg);
    }
}
```

```
void display()
{
    student s1;
    s1.accept();
    s1.display();
    return 0;
}
```

Output :

Enter Name: Rohan

Enter Roll No: 90

Enter class : Sy

Enter Name: Rohan

Enter Roll No: 90

Enter class : Sy

b) Write a C++ program to declare a class book having data members as id, name & price except data for 2 books & display data of book having greater price

```
#include <iostream>
#include <string>
using namespace std;
class book()
{
public:
    string book_name;
    int price, no_of_pages;

    void accept()
    {
        cout << "In enter book name : ";
        cin.ignore();
        getline(cin, book_name);
        cout << "In book price";
        cin >> price;
        cout << "In enter number of pages : ";
    }

    int main()
    {
        book b1, b2;
        b1.accept();
        b2.accept();
```

```
if (b1.price == b2.price)
{
    cout << endl << "Price of both books equals";
}
else if (b1.price > b2.price)
{
    cout << endl << b1.book_name << " has greater price";
}
else
{
    cout << endl << b2.book_name << " has greater price";
}
return 0;
```

Output :

Enter book name : Skigai

Enter book price : 60

Enter number of pages : 200

Enter book name : Sapiens

Enter book price : 70

Enter number of pages : 400

Sapiens has a greater price

c) Write a C++ program to accept time in hours, minutes & seconds for one member & display total time in seconds.

```
#include<iostream>
using namespace std;
class Time {
public:
    int hr, min, sec;

    void accept()
    { cout << "Enter time (HH:MM:SS)" 
        cin >> hr >> min >> sec;
    }

    int convert()
    {
        int secs = (hr * 3600) + (min * 60) + sec;
        return secs;
    }

};

int main()
{
    Time t;
    t.accept();
    cout << endl << "The time in seconds : " << t.convert() <<
    "seconds";
}

return 0;
}
```

*Open
TSP7*

Output

enter time in (HH:MM:ss) : 12

12

12

Total time in seconds is : 43932 :

Practical 2

- a) WAP to declare a class city having data member as name & population. Accept this data for 5 cities and display name of city having highest population.

```
#include<iostream>
#include<string.h>
using namespace std;
class City
{
public:
    int p;
    string name;

    void accept()
    {
        cout << "enter city name : ";
        cin >> ws;
        getline (cin, name);
        cout << "enter population : ";
        cin >> p;
    }

};

int main()
{
    City c[5];
    for (int i=0; i<5; i++)
    {
        cout << "City " << i+1 << ": \n";
        c[i].accept();
    }
}
```

```
int max = 0;  
for (int i=1; i<5; i++)  
{  
    if (c[i].p > c[max].p)  
    {  
        max = i;  
    }  
}
```

```
cout << "In city with highest population : " << c[max].name;  
cout << "Bhavesh Kanjwani";  
cout << "PRN No. S1262240267";
```

return 0;

b) WAP to declare class 'Account' having data members as Account No. & balance.. Accept data for 10 students accounts & give interest of 10% where balance is equal or greater than 5000 and display them.

```
#include <iostream>
using namespace std;
class Account
{
    int acc_no;
    float b;
public:
    void accept()
    {
        cout << "enter account number : ";
        cin >> acc_no;
        cout << "enter balance ";
        cin >> b;
    }
    void display()
    {
        cout << "account number : " << acc_no << "Balance : " << b;
    }
};

int main()
{
    Account a[10];
    for (int i=0; i<10; i++)
    {
        a[i]. accept();
    }
}
```

```
for (int i=0; i<10; i++)  
{  
    if (a[i].balance >= 5000)  
    {  
        a[i].balance += a[i].balance * 0.10;  
    }  
  
    int max = 0;  
    for (int i=1; i<10; i++)  
    {  
        if (a[i].balance > a[max].balance)  
        {  
            max = i;  
        }  
    }  
  
    cout << "In Accounts with Balance >= 5000 : "  
    for (i=0; i<10; i++)  
    {  
        if (a[i].balance >= 5000)  
        {  
            a[i].display();  
        }  
    }  
  
    return 0;  
}
```

3 Write a C++ program class "staff" having data members as name & post.

```
#include<iostream>
using namespace std;
class staff:
{
public:
    string p, n;
    void getdata()
    {
        cout << "enter name and post : ";
        cin >> n >> p;
    }
    void display()
    {
        cout << "Name : " << n << "Post : " << p;
    }
};

int main()
{
    int i;
    staff s[5];
    cout << "enter data for 5 people : ";
    for (i=0; i<5; i++)
    {
        cout << "Name " << i+1 << " Post : " << i+1 << endl;
        s[i].getdata();
    }
}
```

```
int m=0;  
for(i=i-0; i<5; i++)  
{  
    if ( s[i].p == 'Mod' )  
    {  
        m = i;  
    }  
}
```

```
(cout << " People who are MOD : " \n;  
s[m].display();  
return 0;
```

3.

Q
-9/7/25

Assignment - 23

1. Write a program to declare class book containing data book title, author name & price.

```
#include <iostream>
#include <string>
using namespace;
int class book
{
    string btitle, bauthor;
    float price;

public:
    void accept()
    {
        cout << "enter author of book: "; cin >> bauthor;
        cout << "enter title of book: "; cin >> btitle;
        cout << "enter price of book: "; cin >> price;
    }

    void display()
    {
        cout << "Name of author is: " << bauthor;
        cout << "Name of book is: " << btitle;
        cout << "Price of book is: " << price;
    }
};

int main()
{
    book b1; p1=&b1;
    p1. accept();
    p2. display();
    return 0;
}
```

2 WAP to declare class student having data member
roll no & percentage by using this pointer

```
#include <iostream>
using namespace std;
class student
```

```
{ float per;
    int rno;
```

```
public:
```

```
void accept()
```

```
{ cout << "enter roll number : " ; cin >> rno;
    cout << "enter percentage : " ; cin >> per;
```

```
}
```

```
void display()
```

```
{
```

```
cout << "roll number : " << rno;
    cout << "In percentage is : " << per;
```

```
}
```

```
};
```

```
int main()
```

```
{ student s;
```

```
s.accept();
```

```
s.display();
```

```
return 0;
```

```
}.
```

c) WAP to demonstrate the use of nested class.

```
#include <iostream>
using namespace std;
class student
```

{

private:

```
string name;
int rno;
```

public:

```
void inputStudentDetails()
```

{

```
cout << "enter student's name : ";
```

```
getline (cin, name);
```

```
cout << "enter roll number : ";
```

```
cin >> rno;
```

}

```
void displayStudentDetails()
```

{

```
cout << "student name : " << name;
```

```
cout << "roll no. : " << rno;
```

}

}

~~class Marks~~

```
i int sci, eng;
```

public:

```
void inputMarks()
```

{

```
cout << "enter marks in science : "; cin >> sci;
```

```
cout << "enter english marks : "; cin >> eng;
```

}

void displaymarks()

{

cout << "marks in science are: " << sci;

cout << "marks in english are: " << eng;

}

}

}

int main()

{

student s1;

student :: Marks m1;

s1. input student details();

m1. input Marks();

s1. display student details();

m1. display Marks();

return 0;

}

Q=
28/8

Practical - 4.

- a) WAP to create two classes result1 & result2 which stores marks of the student. Read the values of marks for both class objects & compute the average of two results.

```
#include<iostream>
using namespace std;
class result1
{
    float marks;
public:
    void accept()
    {
        cout << "Enter marks : ";
        cin >> marks;
    }
    void display()
    {
        cout << "Marks are : " << marks;
    }
    float getmarks()
    {
        return marks;
    }
};

class result2
{
    float marks;
public:
```

```
void accept ()
```

{

```
    cout << "enter marks : " ;
```

```
    cin >> marks ;
```

}

```
void display ()
```

{

```
    cout << "marks are : " << marks ;
```

}

```
float calaverage (result1 &r1)
```

{

```
    float average ;
```

```
    average = (r1.getmarks() + marks) / 2 ;
```

```
    return average ;
```

}

{,

```
int main ()
```

{

```
    result1 r1 ;
```

```
    result2 r2 ;
```

```
    r1.accept () ;
```

```
    r2.accept () ;
```

```
    r1.display () ;
```

```
    r2.display () ;
```

```
    float ans = r2.calaverage (r1) ;
```

```
    cout << "Average of two results of two marks are : " << ans ;
```

```
    return 0 ;
```

}

b) WAP to find greatest number among two numbers from two different classes using friend function.

```
#include <iostream>
```

```
using namespace std;
```

```
class number2;
```

```
class number1
```

```
{
```

```
    int n;
```

```
public:
```

```
    void accept()
```

```
{
```

```
        cout << "enter a number : ";
```

```
        cin >> n;
```

```
}
```

```
    void display()
```

```
{
```

```
        cout << "number is : " << n;
```

```
}
```

```
    cin >>
```

```
    friend int max(number1 &, number2 &);
```

```
};
```

```
class number2
```

```
{
```

~~int n;~~

```
public:
```

```
    void accept()
```

```
{
```

```
        cout << "enter a number : ";
```

```
        cin >> n;
```

```
}
```

```
    void display()
```

```
{
```

```
    cout << "number is : " << n;
```

```
}
```

```
friend int max (number1 &, number2 &);  
};  
int max (number1 &a, number2 &b)  
{  
    if (a.n>b.n)  
    {  
        return a.n;  
    }  
    else  
    {  
        return b.n;  
    }  
},  
int main()  
{  
    number1 n1;  
    number2 n2;  
    n1.accept();  
    n2.accept();  
    n1.display();  
    n2.display();  
    int ans = max(n1, n2);  
    cout<<"The greater number is : "<<ans;  
    return 0;  
}
```

c) WAP for swapping contents of two variables of same class using friend function.

```
#include<iostream>
using namespace std;
class number
{
    int a;
public:
    void accept()
    {
        cout << "enter a number : ";
        cin >> a;
    }
    void display()
    {
        cout << "number is : " n;
    }
    friend int void swapNumbers(number &, number &);
};

void swapNumbers(number &n1, number &n2)
{
    int temp = n1.a;
    n1.a = n2.a;
    n2.a = temp;
}

int main()
{
    number n1, n2;
    n1.accept();
    n2.accept();
```

```
cout << "original numbers are : ";
n1.display ();
n2.display ();
cout << "In numbers after swaping are : ";
n1.display ();
n2.display ();
return 0;
```

}

~~Q~~
~~R~~

d) Create two classes ClassA & class B each with private integer. Write a friend function sum that can access private data of both classes & return sum.

```
#include <iostream>
using namespace std;
class ClassB;
class ClassA;
{ int A;
public:
    void accept()
    { cout << "enter value of A";
        cin >> A;
    }
```

```
friend int sum (ClassA oA, ClassB oB);
}na;
```

```
class ClassB
{ int B;
public:
    void accept()
    { cout << "enter value of B";
        cin >> B;
    }
```

```
friend int sum (ClassA oA, ClassB oB);
```

```
{nb;
int sum (ClassA oA, ClassB oB)
{
    return oA.A + oB.B;
}}
```

```
int main()
```

```
na.accept();  
nb.accept();  
cout << "sum of two values is : " << sum(na,nb);  
return 0;  
}
```

"A to value ratio" > two }

(In A1201, In A2201) must true because

"A to value ratio" > two }

(In A1201, In A2201) must true because

e) WAP that contains a private integer in class Number.
Use friend function swapNumbers (Number &, Number &) to
swap private values of two Numbers objects.

```
#include <iostream>
using namespace std;
class Number
{
    int num;
public:
    void accept()
    {
        cout << "enter a number : ";
        cin >> num;
    }
    friend swapNumbers (Number &, Number &);
} nl, n2;
void swapNumbers (Number &, Number &)
{
    int temp = nl.num;
    nl.num = n2.num;
    n2.num = temp;
}
int main()
{
    nl.accept(); n2.accept();
    cout << "\nnumbers before swaping are : ";
    nl.display(); n2.display();
    swapNumbers (nl, n2);
    cout << "\nnumbers after swaping are : ";
    nl.display();
    n2.display();
    return 0;
}
```

8 Define two classes Box & cube each having private volume. Using friend function findGreater(Box, Cube) determines which object has larger volume.

```
#include<iostream>
using namespace std;
class Cube;
class Box;
{ double v;
public:
    void accept()
    { double l,w,h;
        cout << "enter value of l & w & h ";
        cin >> l >> w >> h;
    } v = l * w * h;
    } friend void greaterVolume (Box, Cube);
} b;
class Cube;
{ double v;
public:
    void accept()
    { double s;
        cout << "enter side length for the cube : ";
        cin >> s;
        v = s * s * s;
    } friend void findGreater (Box, Cube);
} c;
void findGreater (Box, Cube)
{ cout << " Greater volume : " << res << endl;
```

```
int main()
{
    b.accept();
    c.accept();
    find greater(b,c);
    return 0;
}
```

g Create a class Complex with real & imaginary parts as private members. Use a friend function to add two numbers & return result as a new complex number.

```
#include <iostream>
using namespace std;
class Complex
{
    int r, i;
public:
    void accept()
    { cout << "enter real & imaginary parts : ";
        cin >> r >> i;
    }
    void display()
    { cout << r << "+" << i << "i" << endl;
    }
    friend Complex add(Complex c1, Complex c2);
};

Complex add(Complex, Complex)
{
    Complex sum;
    sum.r = c1.r + c2.r;
    sum.i = c1.i + c2.i;
    return sum;
}

int main()
{
    c1.accept();
    c2.accept();
    Complex sum = add(c1, c2);
    cout << "Sum is : " << sum;
    sum.display();
    return 0;
}
```

h. To create a class with private data members: name & three subject marks. Write a friend function calculateAverage() that calculates & displays average marks.

```
#include<iostream>
using namespace std;
class Student
{
    string N; int m[3];
public:
    void accept()
    {
        cout<<"enter name of student : "; cin>>N;
        cout<<"enter three subject marks : "; cin>>m[0]>>m[1]>>m[2];
    }
    friend void calculateAverage(Student s);
}s;
void calculateAverage(Student)
{
    double avg = (s.m[0] + s.m[1] + s.m[2]) / 3;
    cout<<" Student : "<<s.N;
    cout<<" Average marks : "<<avg;
}
int main()
{
    s.accept();
    calculateAverage(s);
    return 0;
}
```

i. Create a class point with private members x & y. Write a friend function that calculates & returns distance between them.

```
#include <iostream>
using namespace std;
class Point
{
    double x, y;
public:
    void accept()
    {
        cout << "enter x & y coordinates : ";
        cin >> x >> y;
    }
    friend double dist (Point p1, Point p2);
}
double dist (Point, Point)
{
    double dx = p1.x - p2.x;
    double dy = p1.y - p2.y;
    return sqrt ((dx*dx) + (dy*dy));
}
int main()
{
    p1.accept();
    p2.accept();
    cout << "distance : " << dist(p1, p2);
    return 0;
}
```

j) Create two classes BankAccount & Audit. Bank account holds private balance information. Write a friend function in Audit that access & prints balance information for auditing.

```
#include<iostream>
using namespace std;
class BankAccount;
class Audit
{
public:
    void pBal(BankAccount);
};

class BankAccount
{
double b; public:
    void accept()
    {
        cout<<"enter bank account balance : ";
        cin>>b;
    }
    friend void Audit :: pBal(BankAccount);
};

void audit :: pBal(BankAccount)
{
    cout<<"Auditing ... Bank account balance : " << ba.b << ;
}

int main()
{
    ba.accept();
    aud.pBal(ba);
    return 0;
}
```

Ch
28/18

Practical - 5

a WAP to find the sum of numbers from 1 to n
using default constructor

```
#include<iostream>
using namespace std;
class sum
{
    int num;
public:
    sum()
    {
        num=4;
    }
    int calsum()
    {
        int i, add=0;
        for(i=1; i<=num; i++)
        {
            add = add + i;
        }
        cout << "The sum of numbers is : " << add;
    }
};

int main()
{
    sum s1;
    s1.calsum();
    return 0;
}
```

- WAP to find sum of numbers from 1 to n using parameterized constructor

```
#include <iostream>
using namespace std;
class sum
{
    int num;
public:
    sum(int a)
    {
        num = a;
    }
    int calsum()
    {
        int i, add = 0;
        for (i = 1; i <= num; i++)
        {
            add = add + i;
        }
        cout << "sum of numbers is : " << add;
    }
};

int main()
{
    sum s1(7);
    s1.calsum();
    return 0;
}
```

- WAP to find sum of numbers from 1 to n using copy constructor.

```
#include <iostream>
using namespace std;
class sum
{
    int num;
public:
    sum(int a)
    {
        num = a;
    }
    sum(int sum& s)
    {
        num = s.num;
    }
    int calsum()
    {
        int i, add = 0;
        for(i=1; i<=num; i++)
        {
            add = add + i;
        }
        cout << "The sum of numbers is " << add;
    }
};

int main()
{
    sum s1(5);
    s1.calsum();
    return 0;
}
```

b) WAP to declare class 'student' having data members as name & percentage. Accept & display data for one object using default constructor.

```
#include <iostream>
using namespace std;
class student
{
    int p; string name;
public:
    student()
    {
        name = "will";
        p = 98;
    }
    void display()
    {
        cout << "Name of student is: " << name;
        cout << "Percentage of student is: " << p;
    }
};

int main()
{
    student s1;
    s1.display();
    return 0;
}
```

- WAP to declare a class 'student' having data members as name & percentage .Accept & display data for one object using parameterized constructor

```
#include<iostream>
using namespace std;
class student
{
```

```
    int p; string n;
public:
```

```
    student (int a, string b)
```

```
    { p = a;
        n = b;
    }
```

```
    void display ()
```

```
    { cout << "Name of student is: " << n;
        cout << "Percentage of student is: " << p;
```

```
}
```

```
int main()
```

```
{ student s1(99, "Jill");
    s1.display();
    return 0;
}
```

- WAP to declare a class 'student' having data members as name & percentage . Accept & display data for one object using copy constructor.

```
#include<iostream>
using namespace std;
```

```
class student
```

```
{ float p, string n;
```

```
public:
```

```
    student (float a, string b)
```

```
{ p=a;
```

```
    n=b;
```

```
}
```

```
    student ( student &s )
```

```
{ p=s.p;
```

```
    n=s.n;
```

```
}
```

```
void display ()
```

```
{ cout << "Name of student is: " << n
```

```
    cout << "Percentage of student is: " << p;
```

```
}
```

```
int main()
```

```
{ student s1(98.6, " jack");
```

```
student s2(s1);
```

```
s2.display();
```

```
return 0;
```

```
}
```

c Define a class 'college' having data members as rno, name, course. WAP using constructor with default value as "Computer Engineering" for course. Accept this data for two objects of class & display the data.

```
#include<iostream>
```

```
using namespace std
```

```
class college
```

```
{ int rno;
```

```
string name, course;
```

```
public:
```

```
 college()
```

```
{ name = "joe";
```

```
rno = 19;
```

```
course = "Computer engineering";
```

```
}
```

```
 college(string a, int b, string c = "Computer engineering")
```

```
{ name = a;
```

```
rno = b;
```

```
course = c;
```

```
}
```

```
 void display()
```

```
{
```

```
 cout << "\n name of is : " << name;
```

```
 cout << "\n roll no is : " << rno;
```

```
 cout << "\n course is : " << course;
```

```
}
```

```
};
```

```
int main()
```

```
{ college c1("jos", 39), c2("ben", 40);
```

```
 c1.display(); c2.display();
```

```
return 0;
```

```
}
```

d. WAP to demonstrate constructor overloading.

```
#include <iostream>
using namespace std;
class department
{
    int nof, nos;
    string n;
public:
    department()
    {
        n = "it";
        nof = 20;
        nos = 80;
    }
    department(string a, int b, int c)
    {
        n = a;
        nof = b;
        nos = c;
    }
    department(int x, int y)
    {
        name = "rahul";
        nof = x;
        nos = y;
    }
    void display()
    {
        cout << "In name is: " << n;
        cout << "In number of faculty is: " << nof;
        cout << "In number of student is: " << nos;
    }
};

int main()
{
```

```
department d1(), d2("william", 80, 60), d3(80, 75);  
d1.display();  
d2.display();  
d3.display();  
return 0;
```

{

On

Topic

Practical - 6

a WAP to implement single inheritance

```
#include <iostream>
using namespace std;
class person
{
protected:
    string name; int age;
};

class student : public person
{
protected:
    int arno;
public:
    void accept()
    {
        cout << "enter name of person : ";
        cin >> name;
        cout << "enter age of person : ";
        cin >> age;
        cout << "enter roll no. : ";
        cin >> arno;
    }

    void display()
    {
        cout << "name of person is : " << name;
        cout << "In age of person is : " << age;
        cout << "In roll no is : " << arno;
    }
};
```

```
int main()
{
    student sl;
    sl.accept();
    sl.display();
    return 0;
}
```

b) WAP to implement multiple inheritance

```
#include <iostream>
using namespace std;
class academic
{
protected:
    float marks;
};
```

```
class sports
{
protected:
    float score;
};
```

```
class result : public academic, sports
```

```
{
protected:
    float total = 0;
```

```
public:
    void accept()
{
```

```
    cout << "enter marks : ";
    cin >> marks;
```

```
    cout << "enter sports score : ";
    cin >> score;
```

```
    total = (marks + score);
```

```
void display()
```

{

```
    cout << "Academic marks are: " << marks;  
    cout << "In sports score is: " << score;  
    cout << "In result is: " << total;
```

}

};

```
int main()
```

{

```
#include <iostream.h>
```

```
r1.accept();  
r1.display();
```

```
return 0;
```

}

c) WAP to implement multilevel inheritance

```
#include <iostream>
```

```
using namespace std;
```

```
class vehicle
```

```
{ protected:
```

```
    string brand; char model[20];
```

```
}
```

```
class car : public vehicle
```

```
{ protected:
```

```
    char vtype[40];
```

```
}
```

```
class electric_car : public car
```

```
{ protected:
```

```
    float battcap;
```

```
public:
```

```
    void accept()
```

```
{
```

```
    cout << "enter car brand : ";
```

```
    cin >> brand;
```

```
    cout << "enter car model : ";
```

```
    cin >> model;
```

```
    cout << "enter vehicle type : ";
```

```
    cin >> vtype;
```

```
    cout << "enter battery capacity : "
```

```
    cin >> battcap;
```

```
}
```

```
void display()
```

```
{ cout << "\n car brand is : " << brand;
```

```
    cout << "\n car model is : " << model;
```

```
    cout << "\n vehicle type : " << vtype;
```

```
    cout << "\n battery capacity : " << battcap;
```

```
{}
```

```

int main()
{
    electric_car el;
    el.accept();
    el.display();
}

```

d) WAP to implement Hierarchical Inheritance

```

#include <iostream>
using namespace std;
class employee
{
protected:
    int eid; string name;
};

class manager : public employee
{
protected:
    string dept;
public:
    manager()
    {
        cout << "enter employee id : ";
        cin >> eid;
        cout << " enter name : ";
        cin >> name;
        cout << "Dept";
        cin.ignore();
        getline(cin, dept);
    }

    void display()
    {
        cout << "eid : " << eid;
        cout << "name : " << name;
        cout << "Dept : " << dept
    }
}

```

```
class developer : public employee  
{ protected:  
    string pl;  
public:  
    void accept()  
    { cout << "enter programing language :";  
        cin >> pl;  
    }  
    void display()  
    { cout << "language :" << pl;  
    }  
};
```

```
int main()  
{  
    manager m;  
    developer d;  
    m.display();  
    d.accept();  
    d.display();  
    return 0;  
}
```

e) WAP to implement following inheritance

```
#include <iostream>
using namespace std;
class student
{
protected:
    long code;
};

class test : public virtual student
{
protected:
    float per;
};

class sports : public virtual student
{
protected:
    string grade;
};

class result :: public virtual student, public virtual test
{
public:
    void accept()
    {
        cout<<" enter college code : ";
        cin>>code;
        cout<<" enter percentage : ";
        cin>>per;
        cout<<" enter grade : ";
        cin>>grade;
    }

    void display()
    {
        cout<<" college code : "<<code;
        cout<<" percentage : "<<per;
        cout<<" grade : "<<grade;
    }
};
```

int main()

{

 result r1;
 r1.accept();
 r1.display();
 return 0;

}

~~Q1~~
~~411~~

Experiment 7

a) WAP to calculate area of laboratory and classroom using function overloading

```
#include <iostream>
using namespace std;
int calarea( int l, int b )
{
    int area ;
    area = l * b ;
    return area ;
}
```

```
int calarea( int a )
{
    int area ;
    area = a * a ;
    return area ;
}
```

```
int main ()
{
    int l, b, a ;
    cout << " enter length of rectangle : " ;
    cin >> l ;
    cout << " enter breadth of rectangle : " ;
    cin >> b ;
    cout << " enter side of square : " ;
    cin >> a ;
```

```
cout << " area of laboratory is : " << calarea ( 3,4 ) ;
cout << " area of classroom is : " << calarea ( 6 ) ;
```

```
return 0 ;
```

b) WAP to calculate sum of 5 float values and 10 int values

```
#include <iostream>
using namespace std;
```

```
class addition
```

```
{ public:
```

```
    int sum (int num[5])
```

```
    { int sum=0, i;
```

```
        for (i=0; i<5; i++)
```

```
        { sum = sum + num[i];
```

```
        }
```

```
    return sum;
```

```
}
```

```
    float sum (float num[10])
```

```
    { int i; float sum=0;
```

```
        for (i=0; i<10; i++)
```

```
        { sum = sum + num[i];
```

```
}
```

```
    return sum;
```

```
}
```

```
};
```

```
int main()
```

```
{ additional;
```

```
    int i, n[5], j;
```

```
    float m[10];
```

```
    cout << "enter 5 numbers:";
```

```
    for (i=0; i<5; i++)
```

```
{
```

```
    cin >> n[i];
```

```
}
```

```
    int total = al.sum(n);
```

cout << " enter 10 numbers : ";

for (j=0 ; j< 10 ; j++)

{

cin >> m[j];

}

float totall = al.sum(m);

cout << "\n sum of 10 float numbers is : " << totall ;

cout << "\n sum of 5 integers is : " << total ;

return 0;

{

c) WAP to implement Unary '-' operator when used with
Object so that numeric data member of class
is negated

```
#include <iostream>
using namespace std;
int main()
{
    class number
    {
        int num;
    public:
        number()
        {
            num=0;
        }
        number(int n)
        {
            num=n;
        }
        void display()
        {
            cout<<"\n number is: "<<num;
        }
        void operator -()
        {
            num=-num;
        }
    };
    int main()
    {
        number n1(2);
        n1.display();
        -n1;
        cout<<" number after negation is: ";
        n1.display();
        return 0;
    }
}
```

Q
U11

Experiment 8

a) WAP to overload the '+' operator so the two strings can be concatenated:

```
#include <iostream>
#include <string.h>
using namespace std;
class mystring
{
    char str[20];
public:
    mystring (char a[20])
    {
        strcpy(str,a);
    }
    void display ()
    {
        cout<<" string is : "<<str<<"\n";
    }
    my string operator +(mystring);
};

mystring mystring::operator +(mystring s2)
{
    mystring s3;
    strcpy(s3.str,str);
    strcat(s3.str,s2.str);
    return s3;
}

int main()
{
    mystring s1("abc"); mystring s2("xyz");
    s3=s1+s2;
    cout<<"\n concatenated ";
    s3.display ();
    return 0;
}
```

b) WAP to create base class Ilogin :-

```
#include <iostream>
using namespace std;
class ilogin
{
protected:
    char n[20], p[20];
public:
    virtual void accept()
    {
        cout << "enter name : ";
        cin >> n;
        cout << "enter password : ";
        cin >> p;
    }
    virtual void public()
    {
        cout << "\n name is : " << n;
        cout << "\n password is : " << p;
    }
};

class email_login : public ilogin
{
public:
    void accept()
    {
        cout << " enter email id : ";
        cin >> n;
        cout << " enter email-id password : ";
        cin >> p;
    }
    void display()
    {
        cout << "\n email id is : " << n;
        cout << "\n email id password is : " << p;
    }
};
```

```
class membership_login : public ilogin
{
public:
    void accept()
    {
        cout << "enter membership name: ";
        cin >> n;
        cout << "enter membership password: ";
        cin >> p;
    }

    void display()
    {
        cout << "In membership name is: " << n;
        cout << "In membership password is: " << p;
    }
};

int main()
{
    ilogin *il, il1;
    email_login el;
    membership_login ml;
```

int type;
cout << "\n 1:- for email login \n 2:- for membership login \n 3:-
normal login :";
cin >> type;

if(type == 1)
{
 il = ⪙
 il->accept();
 il->display();
}

else if(type == 2)
{
 il = &ml;
 il->accept();
 il->display();
}

```
else if (type == 3)
{
    il = & ill;
    il-> accept();
    il-> display();
}
return 0;
```

~~Qn
n/1~~

Experiment 9.

1. WAP to copy the contents of one file to another

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main()
{
    ifstream f1;
    ofstream f2;
    char ch;
```

```
f1.open((ch = f1.get()) != EOF);
f2.open("abc.txt");
```

```
f1.open("myfile.txt", ios::in);
f2.open("abc.txt", ios::out);
```

```
while ((ch = f1.get()) != EOF)
```

```
{
```

```
    f2.put(ch);
}
```

~~```
f1.close();
f2.close();
```~~~~```
cout << "contents copied : ";
```~~~~```
return 0;
```~~

{

b) WAP to count digits and spaces using file handle

```
#include <iostream>
#include <fstream> #include <cctype.h>
using namespace std;
int main()
{
 ifstream fl("myfile.txt");
 char ch;
 int d=0, s=0;

 while ((ch=fl.get())!=EOF)
 {
 if (isdigit(ch))
 {
 d++;
 }
 else if (isspace(ch))
 {
 s++;
 }
 }

 cout<<"\n number of digits in file are: "<<d;
 cout<<"\n number of spaces in file are: "<<s;
 return 0;
}.
```

c) WAP to count words using file handling.

```
#include <iostream>
#include <fstream>
#include <ctype.h>
using namespace std;
int main()
{
 ifstream f1("myfile.txt");
 char ch;
 int w=0;
 while ((ch=f1.get())!=EOF)
 {
 if(isspace(ch))
 {
 w++;
 }
 }
 cout<<"total words are : "<<w;
 return 0;
}
```

d) WAY TO COUNT OCCURENCES OF A GIVEN WORD  
USING FILE HANDLING

```
#include <iostream>
#include <fstream> #include <ctype.h>
using namespace std;
```

```
int main()
```

```
{
```

```
ifstream f1("myfilee.txt");
```

```
string ch;
```

```
int w=0;
```

```
cout<<"enter a word:";
```

```
cin>>ch;
```

```
string word;
```

```
while(f1>>word)
```

```
{
```

```
cout<<"In "<<word;
```

```
if(ch==word)
```

```
{
```

```
w++;
```

```
g
```

```
}
```

```
f1.close();
```

```
cout<<"Occurrence of a word in file is : "<<w;
```

```
return 0;
```

```
},
```

~~Ques~~  
~~1/1~~

## Experiment 10

i WAP to find sum of array using function template

```
#include <iostream>
```

```
using namespace std;
```

```
template <class T>
```

```
T sum (T arr [10])
```

```
{ int i;
```

```
 T add=0;
```

```
 for(i=0; i<10; i++)
```

```
{
```

```
 add=add+arr[i];
```

```
}
```

```
 return add;
```

```
}
```

```
int main()
```

```
{ float arr [] = {12, 1, 3, 5, 6, 3, 4, 9, 10, 14, 15};
```

```
cout << "Insum of array elements is : " << sum(arr);
```

```
return 0;
```

```
}
```

ii Write a C++ program of square function using template specialization

```
#include <iostream>
using namespace std;
template <class t> t square (t num)
{
 return num * num;
}
template <> string square (string str)
{
 return str * str;
}
int main()
{
 int n = 5;
 string t = "hello";
 cout << "square of no. is: " << square (n);
 cout << "square of string is: " << square (t);
 return 0;
}
```

iii) WAP to build simple calculator using class Template

```
#include <iostream>
#include <math.h>
using namespace std;
template < class T > class Cal
{
 T a, b;
public:
 Cal(T c, T d)
 {
 a = c;
 b = d;
 }
 T add()
 {
 return a+b;
 }
 T subt()
 {
 return a-b;
 }
 T div()
 {
 return a/b;
 }
 T mult()
 {
 return a*b;
 }
 T Sqrt()
 {
 return sqrt(a);
 }
 T Cbrt()
 {
 return cbrt(a);
 }
}
```

```
t Pow()
{
 return pow(a);
}

t Sin()
{
 return sin(a);
}

t Cos()
{
 return cos(a);
}

t Tan()
{
 return tan(a);
}

};
```

```
int main()
{
 int a, b, ch;
 cout << "enter two numbers ";
 cin >> a >> b;
 Cal<float> cl(a, b);
 do {
```

~~\n1: to add \n2: subtract \n3: divide \n4: multiply \n5: square  
\n6: Cuberoot \n7: Power \n8: Sin \n9: Cos \n10: Tan  
\n11: exit \n" enter your choice : "~~

```
scanf("./d", &ch);
```

```
switch (ch)
```

```
{ case 1:
```

```
 cout << "addition is: " << cl.add();
```

```
 break;
```

```
case 2:
```

```
 cout << "Subtraction is: " << cl.subt();
```

```
 break;
```

```
case 3:
```

```
 cout << "Division is: " << cl.div();
```

```
 break;
```

case 4:

cout << "multiplication is: " << cl.mult();  
break;

case 5:

cout << "square root is: " << cl.sqrt();  
break;

case 6:

cout << "cube root is: " << cl.cbrt();  
break;

case 7:

cout << "sin is: " << cl.sin();  
break;

case 8:

cout << "cos is: " << cl.cos();  
break;

case 9:

cout << "tan is: " << cl.tan();  
break;

case 10:

cout << "Power is: " << cl.pow();  
break;

}

} while(ch != 11);  
return 0;

}.

Ques  
5/11

## Experiment 11

a WAP to modify the value of given element

```
#include <iostream>
using namespace std;
template <typename T> class Vec
{
 T* v;
 int size;
public:
 vec(int n): size(n)
 {
 v = new T [size];
 }
 void set (int idx, T val)
 {
 v[idx] = val;
 }
 void mult()
 {
```

```
 for(int i=0; i<size ; i++)
 {
 v[i]*=2;
 }
```

```
 void display()
```

```
 cout << "(";
 for(int i= 0; i<size ; i++)
 {
 cout << v[i];
 if(i != size -1)
 {
 cout << ",";
 }
 }
```

{,

int main()

{ int n;

cout &lt;&lt; "enter no of elements : ";

cin &gt;&gt; n;

vec &lt;int&gt; v(n);

for (int i=0; i&lt;n; i++)

{ cout &lt;&lt; "enter value for element " &lt;&lt; i+1 &lt;&lt; ":";

int val;

cin &gt;&gt; val;

v.set(i, val);

{

v.display();

int idx, newVal;

cout &lt;&lt; "enter index to modify : " &lt;&lt; n-1 &lt;&lt; ":";

cin &gt;&gt; idx;

cout &lt;&lt; "enter new value : " &lt;&lt; idx;

cin &gt;&gt; newVal;

v.set(idx, newVal);

cout &lt;&lt; "vector after multiply : " ;

v.display();

return 0;

{.

case 3:

```
if(v.empty())
{
 cout << "stack underflow";
}

else
{
 cout << "In stack is: ";
 stack<int> temp=v;
 while (!temp.empty())
 {
 cout << "temp.top() << " " ";
 temp.pop();
 }
 break;
}
```

case 4:

```
cout << "bbyee";
break;
}
} while(ch!=4);
return 0;
}
```

## Experiment 12

i WAP using STL to implement stack.

```
#include <iostream>
#include <stack>
using namespace std;
int main()
{
 stack<int> v;
 int ch, n, data, i;
 do
 {
 cout << "1: to push\n2: to pop\n3: to display\n4: to exit
 \nEnter your choice";
 cin >> ch;
 switch (ch)
 {
 case 1:
 cout << "enter data to push ";
 cin >> data;
 v.push (data);
 break;
 case 2:
 if (v.empty ())
 {
 cout << "queue overflow ";
 }
 else
 {
 v.pop ();
 cout << "data popped : ";
 }
 break;
 }
 } while (ch != 4);
```

case 3:

```
if(v.empty())
{
 cout << "stack underflow:";
```

else

```
{ cout << "In stack is: ";
 stack <> int > temp = v;
 while (!temp.empty())
 {
 cout << *temp.top() << " ";
 temp.pop();
 }
 break;
}
```

Case 4:

```
cout << "bbyee";
break;
}
} while(ch != 4);
return 0;
}
```

Ques  
|||||