1) Arithmatic operations

```
# include <iostream.h>
void main ()
{
 cout <<"Enter '+' for addition \n";
 cout <<"Enter '-' for substraction \n";
 cout <<"Enter '*' for multiplication \n";
 cout <<"Enter '/' for division \n";
 int a, b;
 Cout <<"Enter the value of a and b";
 Cin >> a >> b;
 char ch;
 Cout <<"Enter your choice";
 Switch (ch)
 {
  case '+' : int sum;
  sum = a+b;
 Cout << Addition = " << sum;
 break;
 Case '-' : int sum;
  sum = a-b;
  Cout << substraction = " << sub";
  break;
  Case '*' : int mul;
  mul = a*b;
  cout << multiplication = " << mul";
  break;
  case '/' : int div;
  if (b = = 0)
  cout <<" Error, division by 0";
  else
 {
```

```
div=a/b;
cout<<"Division = "<<div;
break;
default: Cout << " Enter elements correctly";
break;
}
return 0;
}
```

2)
```
#include <iostream.h>
using namespace std;
int main()
{
int n;
cout<<"Enter a number \n";
cin>>n;
if (n%2==0)
cout<<"Number is even \n";
cin>>n;
if (n%2==0)
cout<<"number is even \n";
else
cout<<"Number is odd \n";
return 0;
}
```

3)
```
#include <iostream.h>
using name space std;
int main()
{ Cout<<"Numbers from 1 to 10 are printed \n";
for(int i=1;i<=10;i++)
{
cout<<i<<"\n";
}
return 0;
}
```

4)
```
#include <iostream.h>
using namespace std;
int main ()
{
int i=1;
while (i<=10)
{
cout<<i<<"\n";
i++;
}
return 0;
}
```

A)5)

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

```
#include <iostream.h>
using namespace std;
int main ()
{
int rows=3;
{
int rows=3;
for (int i=1;i<=rows;i++)
{
for (int j=1;j<=i;j++)
{
cout<<"*";
```

```
}
cout <<endl;
}
return 0;
}
```

B)
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
#include <iostream.h>
using namespace std;
int main ()
{
 int main()
 {
 int r=5;
for(int i=1 ;i<=r; i++)
{
 for(int j=1; j<=i ;j++)
{
 Cout<<i<<" ";
}
 cout <<endl;
}
 return 0;
}
```

C)
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
#include <iostream.h>
using namespace std;
int main ()
{
  int r=5;
for (int i=1 ; i<=r ; i++)
{
 for (int j=1 ; j<=i ; j++)
{
  cout <<i<<" ";
}
 cout <<endl;
}
 return 0;
}
```

→ Experiment-1.

1) Write a C++ program to declare a class student having data members as roll no. and name. Accept and display data for single student.

```
#include <iostream>
using namespace std;
class student
{
 int roll_no;
 string name;
 public:
  void accept ()
  {
   cout <<"Student Name :" << name ;
   cout <<"\n Student Roll No." <<roll_no;}};
   cin >> name >> roll no.;}
  void disp ()
  {
   cout << " Student Name:" <<name;
   cout <<" \n Student Roll No:" <<roll_no;}};
   int main ()
   {
    student S1;
    S1. accept ();
    S1. disp ();
    return 0;
   }
```

output:-
Enter the student name and roll no: Durwanki 63
Student Name : Durwanki
Student Roll No: 63.

2) Write a C++ code to create a class book having data members as book_name, book_price and book-id. Accept the data for 2 books and display the name of the having greater price.

```
#include <iostream>
using namespace std;
class class_book
{
 string book-name;
 int book_id;
 public:
 int book_price;
 void accept ()
  {
  cout <<" Enter the book name, price and ID:";
  cin>> book_name >>book_price>>book-id;
 }
 void disp ()
 {
  cout <<"\n Book Name:" book_name;
  cout <<"\n Book Price:" book_price;
  cout <<" \n Book ID:" book_id;
 }};
 int main ()
 {
  class_book_price B1, B2;
  B1. accept ();
```

```
B2.accept();
if (B1.book_price > B2.book_price)
{
 B1.disp();}
 else
 {
  B2.disp();
 }
 return 0;
}
```

2) WAP to declare a class time having data members as H,M,S. Accept data for one object and display total time in seconds.

```
#include <iostream>
using namespace std;
class time;
{
 Public:
  int H,M,S;
  void accept(){
   cout<< "Enter time in Hours, min, sec";
   cin>> H>>M>>S;
  }
  void disp(){
  t=(h*3600)+(m*60)+S;
  cout << "Total time" << t;
  }
  };
  int main(){
  Time t1
  t1.accept()
```

```
t1.disp()
return 0;
}
```

output:-
Enter the time in hours, min, sec:
4:32:60

Total time: 16,380 sec

→ Experiment-2

1) WAP to declare a class city have data members as name and population. Accept this data for 5 cities and display name of the city having highest population.

```
#include <iostream>
using namespace std;
class city{
Public:
  int p;
  string n;
void accept ){
cout<<"enter name and population";
  cin>>n>>p;
}
};
int main(){
city C[5];
for (int i=0 ;i<5; i++){
C[i]. accept ();
}
  int main = C[0]p;
  for (int i=0; i<5; i++){
  if (C[i].p >max){
    max=i
}
}
cout << " city population" << C[max].n;
return 0;
}
```

output:- ~~name~~
enter name and population: Mumbai 25
" " Pune 45
" " Kolkata 70
" " Jaipur 23
" " Raipur 75

city population: Raipur

2) WAP program to declare a class 'Account' having data ~~of~~ members as number and balance. Accept this data for 10 accounts and give interest of 10% were balance is equal or greater than 5000 and display them.

```
#include <iostream>
using namespace std;
class account {
Public:
in. float B;
int N;
void accept ){
cout <<"Enter account number and balance";
  cin>>n>>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].b>=5000){
a[i].b = a[i].b* 0.1 + a[i].b;
}
}
cout << " account balance "<<a[i]b;
return 0;
```

output:-

```
Enter Account Number : 2564
Enter Account Balance : 2094
———"——— : 4879
———"——— : 7890
———"——— : 6789
———"——— : 4567
———"——— : 3578
———"——— : 2478
———"——— : 3678
———"——— : 2634
———"——— : 3678
———"——— : 2634
———"——— : 9705
———"——— : 5738
———"——— : 57228
———"——— : 5835
———"——— : 3789
———"——— : 4627
———"——— : 8906
———"——— : 7568
———"——— : 4689
Account Number: 7890
Account Balance : 7467.9
Account Number : 2634
Account Balance : 106755
Account Number : 5738
Account Balance : 62950.8
```

3) WAP to declare a class 'Staff' having data members as name and post. Accept this data for 5 staff and display names of staff who are "HOD".

```
#include <iostream>
using namespace std;
class staff
{
string name;
string Post;
Public:
void ~~display~~ accept()
{
cout << "Enter staff name";
cin>>name;
cout << " Enter Staff post";
cin >> post;
}
void display()
{
cout << "Staff name :" << name;
cout << " Staff post :" << post;
}
}
```

```
int main()
{
 staff S[5];
 int i;
 for (i=0; i<5; i++)
  {
    S[i].accept();
  }
 for(i=0; i<5; i++)
  {
  if (S[i].post==="HOD")
  {
    S[i].display();
  }
  }
 return 0;
}
```

output:-

```
Enter staff name: A
Enter staff post: Lecturer
Enter staff name: B
Enter staff post: HOD
Enter staff name: C
Enter staff post: Manager
Enter staff name: D
Enter staff post: CEO
Enter staff name: E
Enter staff post: HOD
Staff name: B
Staff post: HOD
Staff name: E
Staff post: HOD
```

→ Experiment-3

1) WAP to declare a class 'book' containing data members as book title, author_name and price. Accept and display the info for 1 object using pointer to that object.

```
#include <iostream>
using namespace std;
class book
{
 string title;
 string author_name;
 int price;
public:
 void accept()
{
 cout<<"Enter book title";
 cin>>btitle;
 cout<<"Enter author name";
 cin>>author name;
 cout<<"Enter book price";
 cin>>price;
}
void display()
{
 cout<<"Book title:"<<btitle;
 cout<<"Author name:"<<author_name;
 cout<<"Book price:"<<price;
 }
 };
 int main()
 {
```

```
book b1;
book *p;
p = &b1;
p -> accept();
p -> display();
}
```

output:-
Enter book title: Clglife
Enter Author Name : Durwanki
Enter price of book : 2500
Book title: ~~Durwanki~~ Clglife
Author name :- Durwanki
Book price : 2500

b) WAP to declare a class 'student' having data members roll_no and percentage. Using 'this' pointer invoke member functions to accept and display this data for one object of class.

```
#include <iostream>
using namespace std;
class student
{
 int roll_no;
float p;
public:
void accept ()
{
 cout <<" Enter Roll_no:";
 cin>> this -> roll_no;
 cout<< "Enter percentage:";
 cin>> this -> p;
}
 void display ()
 {
  this -> accept ();
 cout << "Roll no." << roll_no;
 cout << " Percentage " << p;
}
};
int main ()
{
 student S1;
 S1.display();
 return 0;
}
```

output:-
Enter Roll_no: 72
Enter percentage : 100
Roll_NO: 72
Percentage : 100

c)
```
#include <iostream>
using namespace std;
class student
{
 Public:
 String name;
int roll;
void accept()
{
cout << " Enter the name of student \n";
cin>> name;
```

```
cout<<" Enter the roll no.\n";
cin>>roll ;
}
void display()
{
cout <<" The name of student" <<name;
cout <<" The roll no.\n" <<roll;
}
class marks
{
Public:
int m1, m2;
void accept 1()
{
cout <<" Enter marks of two subject"
cin>> m1 >> m2;
}
void display 1()
{
float per = ((m1+m2)/100.0)* 100;
cout <<" The percentage of 2 subjects is:"
<< per <<endl;
}
};
};
int main()
{
student s1;
student :: marks m;
s1.accept();
m.accept 1();
s1.display();
m.display1();
return 0;
}
```

output:-

```
Enter the name : Durwanki
Enter roll no: 63
Enter the name : Durwanki
The roll no: 63
Enter two marks:
32
16
Percentage : 48
```

→ Experiment-4

a) WAP to swap two numbers from same class using object as function argument. Write swap function as member function.

```
#include <iostream>
using namespace std;
class numbers {
private:
  int a,b;
public:
int temp;
void accept(){
cout <<" Enter the first number:";
cin>>a;
cout <<"Enter the second number:";
cin>>b;
}
void display(){
cout<< \n After swapping:"<<endl;
cout <<"First number="<< a <<endl;
cout <<" Second number="<<b<< endl;
}
void swap(numbers &n)
{
n.temp=n.a;
n.a=n.b;
n.b=n.temp;
}
int main(){
numbers n;
n.accept();
n.swap(n);
n.disp();
return 0;
}
```

output:-
Enter the first number:9
Enter the second number:2
After swapping:
First number:2
Second number:9

b) WAP to swap two numbers from same class using concept of friend function.

```
#include <iostream>
using namespace std;
class number {
private:
 in a,b, temp;
public:
void accept (){
cout <<" Enter the first numbers:";
cin >> a;
cout <<"Enter the second number:";
cin>>b;
}
void display (){
cout <<"\n After swapping:"<<endl;
cout <<"First number="<<a<<endl;
cout<<"Second number="<<b<<endl;
}
friend void swap (number &);
```

```
};
void swap (number & n){
n.temp = n.a;
n.a = n.b;
n.b = n.temp;
}
int main()
{
number n;
n.accept();
swap (n);
n.display();
return 0;
}
```

output:-
Enter the first number: 7
Enter the second number: 4
After swapping
First number: 4
Second number: 7

c) WAP to swap two numbers from different class using friend function.

```
#include <iostream>
using namespace std;
class num 2;
class num 1{
private:
int a;
public:
void accept(){
cout << "Enter the first number:";
cin >> a;
}
friend void swap (num1 & num2 &);};
class num 2{
private:
int b;
public:
void accept(){
cout << "Enter second number:";
cin >> b;
}
friend void swap(num1 &, num2 &);
};
void swap (num1 & x, num2 & y){
int temp;
temp = x.a;
x.a = y.b;
y.b = temp;
cout << "After swapping:" << endl;
cout << "First number =" << x.a << endl;
cout << "Second number =" << y.b << endl;
}
int main(){
num1 n1;
num 2 n2;
n1.accept();
n2.accept();
swap (n1, n2);
return 0;
}
```

output:-

Enter the first number = 5
Enter the second number = 6
After swapping:-
First number = 6
Second number = 5.

d) WAP to create two classes Result 1 and Result 2 which stores the marks of the students. Read the value of marks for both the class object and compute the average of two results.

```
#include <iostream>
using namespace std;
class result 2;
class result 1{
private:
string name;
float marks;
Public:
void accept(){
cout << "Enter student name:";
getline (cin, name);
cout << "Enter the total marks obtained in first semester out of 100:";
cin >> marks;
}
friend void average (result 1, result 2);
};
class result 2 {
private:
float marks;
public:
void accept (){
cout << " Enter the total marks obtained in the second semester out of 100:";
cin >> marks;
}
friend void average (result 1, result 2);
};
void average (result 1 X, result 2 Y){
float avg;
avg = (X.marks + Y.marks) /2;
cout << "Avg of both the results" << avg;
}
int main(){
result 1 r1;
result 2 r2;
r1.accept();
r2.accept();
average (r1, r2);
return 0;
}
```

output:-
Enter student name: Durwanki Satpute
Enter the total marks obtained in first semester out of 100: 98
Enter the total marks obtained in second semester out of 100: 96
Avg. of both the results = 97.

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

```
#include <iostream>
using namespace std;
class number 2;
class number 1{
int a;
Public:
void input(){
cout<<"Enter number for classes:";
cin>>a;
}
friend void greatest(number n1, number n2);
};
class number 2{
int b;
Public:
void input(){
cout<<"Enter number for class 2:";
cin>>b;
}
friend void greatest(number n1, number n2);
};
void greatest(number n1, number2 n2){
if (n1.a>n2.b)
cout<<"greatest:"<<n2.b<<endl;
}
int main(){
number 1 obj 1;
number 2 obj 2;
obj.input();
obj.input();
opeatest(obj 1, obj 2);
return 0;
}
```

output :-

```
Enter number for class 1: 45
Enter number for class 2: 67
Greatest number: 67
```

→ Friend function (Practice)

1) Create two classes A and B with a private integer. Write a friend function sum() that can access private data from both the classes and return the sum.

```
#include <iostream.h>
using namespace std;
class class B;
class class A;
int A;
public:
void accept(){
cout<<"Enter value for A:";
cin>>A;
}
friend int sum (class A oA, class B (B);
} na;
class class B{
int B;
Public:
void accept ()}
cout <<" Enter value for B:";
cin>>B;
}
friend int sum / class OA, class OB){;
}nb;
int sum (class OA, class B OB){
return OA.A+OB.B;
}
int main(){
```

```
na.accept();
nb.accept();
cout <<"Sum:" <<Sum (na,nb) <<endl;
return 0;
}
```

Output :-

Enter value for A:3
Enter value for B:6
Sum:9

2. Write a program with a Class Number that contain a private integer. Use a friend function Swapnumber (Number &, number &) to swap the private values of two number objects.

→

```
#include <iostream>
using namespace std;
class number{
int main;
Public;
void accept ){
Cout <<"Enter value:";
cin>>num;
}
void display ){
Cout <<"Value":<<num<<endl;
}
friend void snapnumber (number &num1,
number &num2);
```

```
3 n1, n2;
void swap numbers (number& , number&){
    int temp = n1.num;
    n1.num = n2.num;
    n2.num = temp;
}
int main (){
    n1.accept();
    n2.accept();
cout << "Before swap:" << endl;
    n1.display();
    n2.display();
swap numbers (n1, n2);
cout << "After swap:" << endl;
    n1.display();
    n2.display();
return 0;
}
```

Output :-

Enter value : 3
Enter value : 6
Before swap
Value : 3
Value : 6
After swap ;
Value : 6
Value : 3

1. Define two classes Box and Cube ; each having a private volume. Write a friend function find greater (Box, Cube) that determines which object has a larger volume.

→

```
#include <iostream>
using namespace std;
class Cube ;
  Class Box
{
  double v;
  public ;
   void accept () {
  double w, l, h;
  cout << "Enter length, width, & height of the
box :";
  cin >> l >> w >> h;
   v = l* w* h ;
}
  friend void findgreater (Box b, Cube c);
} 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 B, cube C);
};
width findGreater (Box, Cube) {
  string xs = (b.v > c.v) ? "Box" : ((c.v > b.v) ?
                "cube" : "Both equal");
  cout << "Greater volume" << xs << endl;
}
int main () {
    b.accept () ;
    c.accept () ;
  find greater (b,c);
  return 0;
}
```

Output :-

Enter l, w, h for Box : 3, 4, 3.
Enter side length for the cube : 3.
Greater volume = Box.

4. Create a class complex with real & imagery parts as a private member use a friend function & add two complex no's and return the result as a new complex ~~digit~~ object.

→

```
# include <iostream>
using namespace std;
class complex {
double r, i;
public :
void accept () {
cout << "Enter real and imajery parts :";
cin >> r >> i;
}
void display () {
cout << r << " + " << i << " i" << endl;
}
friend complex add (complex c1, complex c2);
} c1, 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 << " The sum is : " ;
sum.display () ;
return 0 ;
}
```

5. Create a class student with private data members & three subject marks. Write a friend function calculate (Average) (student) that calculate the avg marks.

→

```
#include <iostream>
using namespace std;
Class Student {
  String n;
  int m[3];
Public :
  void accept ()
{
  cout << " Enter student name : " ;
  cin >> n;
  cout << " Enter marks from three subject";
  cin >> m[0] >> m[1] >> m[2] ;
}
  friend void calculate Average (Student) {
  double avg = (s.m[0] + s.m[1] + s.m[2])
  cout << " Student : " << s.n << endl;
  cout << " Average Marks : " << avg <<
endl;
}
  int main () {
  s. accept ();
  calculate Average (s) ;
  return 0;
}
```

8) Create three classes Alpha, Beta, Gamma each with private data members. With a single friend function that can access all three & prints their sum.

→

```
int n; public:
  void accept ()
{
  cout << " Enter value for Alpha : " ;
  cin >> n;
}
  friend int sum (Alpha a, Beta b, Gamma c);
} a;
  Class Beta {
  int n; public;
  void accept () {
  cout << " Enter value for beta : " ;
  cin >> n;
}
  friend int sum (Alpha a, Beta b, Gamma c)
} c;
  int sum (Alpha a, Beta b, Gamma c) {
  return a.n + b.n + c.n ; }
int main () {
  a. accept ();
  b. accept ();
  c. accept ();

  cout << " Sum " << sum (a, b, c) << endl;
  return 0;
}
```

8) Create a class Point with private members x & y. Write a friend function that calculates and returns the distance between the two points.

```
→ # include <iostream>
  using namespace std;
# include <cmath>
Class points {
double x, y;
Public :
void accept () {
cout << "Enter x & y coordinates" :;
cin >> x >>y
}
 friend double dist (Point p1, Point p2);
} p1, 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) << endl;
return 0;
}
```

8. Create the classes : ~~Bankaccount and Audit~~ ~~Bankacc~~ Bankaccount and Audit. Bankaccount hold private banau info. Write a friend function in Audit that access & prints balance information for auditing.

```
→ # include <iostream>
using namespace std;
class Bank Account:
class Audit {
public;
  void pBal (BankAccount);
} and ;
   class BankAccount {
  double b; public:
  void accept()
{
 cout << "Enter Bank acc Balance:";
 cin >> b;
}
 friend void Audit :: PBal (BankAcc);
} ba;
  void Audit :: PBal (BankAccount)
{
 cout << "Auditing Bank Acc balance : ₹"
    << ba.b << endl;
}
 int main () {
 ba.accept ()
 and.pBal (ba);
 return 0;
}
```

Ph

Page No.:
Date:

# Experiment - 5.

Q1- Write a programe to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

→

```
# include(iostream)
using namespace std;

class Sum
{
  int n, total;
Public:
  Sum(int num){
    n= num;
    total = (n*(n+1))) /2;
}
  void display ()
  cout << "Sum of number from 1 to n" <<n<<
  " = "    <<total <<endl;
}
};
int main (){
  int n;
cout <<"Enter value of n:";
cin >>n;

Sum obj (n);
obj. display ()

return 0;
```

Page No.:
Date:

```
}
```

b) Write a program to declare a class "student" having data members as name and percentage. Write a constructor to initialize these data members. Accept and display data for one students.

→

```
# include <iostream>
using namespace std;
class student
{
  string name;
  float p;
  Public:
  student (string name, float p)
{
  name=n;
  percentage=p;
}
  void display ()
  cout << "Student name = " <<name<<endl;
  cout << " Percentage = "<<p <<endl;
}
};
int main ()
  student S1 ("Aaryan",90);
  S1.display ();

{ cout << " Enter Student name :";
  cin>> name;
```

cont...

```
{ cout << "Enter percentage : " ;
  cin >> p;
```

• WAP to find the sum of number 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 << " \n Sum of number is : "
  << add ;
  }
};
  int main()
  {
    sum s1 ;
  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 << " \n sum of numbers is " << add;
  };
  int main ()
  { sum s1 (s);
    s.1 . 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 = "Aryan"
  p=98;
}
  void display()
{
  cout << " In name of student is : " << name;
  cout << " In 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 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 << " In name of student is : " << n;
  cout << " In percentage of student is : " << p;
}
};
int main()
{ student s (98.6, "jack");
  student s2 (s1)
  s2.display();
  return 0;
}
```

c) Define a class 'college' data member as roll no., name, course. WAP using constructor with default value as "Computer Engineering" for course. Accept this data for two objects of class and display the data.

```
#include <iostream>
using namespace std;
class college
{
 int roll no.;
string name, course;
public:
 college ()
{
 name = "Purnwanti";
roll_no = 63;
course = "Computer Engineering";
}
 college (string a; int b, string c="Computer Engineering
 {
  name = a;
  roll_no = b;
  course = c;
 }
 void display ()
 {
  cout <<"\n name is:" << name;
  cout <<"\n rollno is:" << roll_no;
  cout <<"\n course is:" << course;
 }
};
```

```
int main ()
{
 college C1 ("Purnwanti", 63), C2 ("Tanisha", 13);
   C1. display ();
   C2. display ();
   return 0;
}
```

d) WAP to demonstrate constructor overloading.

```
#include <iostream>
using namespace std;
class department
{
 int not, 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 <<"\n name is :"<< n;
 cout <<"\n number of faculty is:"<<nof;
 cout <<"\n number of student is:"<<nos;
}
};
int main ()
{ department d1 ();
d2(" willam",80,60),
d3(80,75),
d1.display();
d2.display();
d3.display();
return 0;
}
```

Experiment-6.

i) Single Inheritance.
Create a base class called person with attributes name and age. Derive a class student from person that adds an attribute rollNumber. Write function to display all details of the student.

```
#include <iostream>
using namespace std;
class persons
{
 string n;
 int a;
Protected;
 void accept ()
{
 Cout<<" Entername :";
 cin>> n;
 cout <<" Age :" <<a;
 }
};
Class student : Public person
{
 int roll;
Public:
 void accept () { person::accept ();
 cout <<" Enter roll no :";
 cin >> roll;
}
 void display ()
{ person :: display ();
```

```
cout<<"\n Roll no :" <<roll;
}
};
int main()
{
student s;
s.accept();
s.display();
return 0;
}
```

output:-
Enter name : Durwanki
Enter age : 20
Enter roll no : 63
Name : Durwanki
Age : 20
Roll no : 63

ii) Multiple Inheritance.
Create two base classes Academic and sports.
- Academic class contains ~~sports~~ ~~score~~. marks of a student.
- Sports class contains sports score.

Create a derived class Result that inherits from both Academic and sports. Write a function to calculate the total score and display details calculate the total score and display details.

```
#include <iostream>
using namespace std;
class academic
{
protected:
int marks;
public:
void accept in()
{
cout <<"Enter marks:";
cin>> marks;
}
};
class sports
{
protected:
int score;
public:
void accepts()
{
cout <<"Enter score:";
cin >> score;
}
};
class result: public academic, public sports
{
public:
void display()
{
int total;
total = marks+ score;
cout <<"\n Marks:" <<marks;
cout <<"\n Score:" <<score;
cout <<"\n Total:" <<total; }};
```

```
int main()
{
  result r;
  r. accept m();
  r. accept s();
  r.display();
  return 0;
}
```

Output:-
Enter marks:30
Enter Score: 50
Marks:30
Score :50
Total :80

iii) Multilevel Inheritance
Create a class vehicle with attributes like brand and model. Derive a class Car from vehicles which adds an attribute type, Further derive a class ElectricCar from car which adds battery capacity. Write function to display all the details.

```
#include<iostream>
using namespace std;
class vehicle {
 protected:
  string brand;
  string model;
 Public:
  vehicle (string b, string m){
   brand=b;
   model=m;
  }
  void display vehicle(){
  cout <<"Brand :" <<brand <<endl;
  cout <<"Model: " <<model <<endl;
  }
};
class Car : public Vehicle {
protected:
 string type;
public:
 Car (string b, string m, string t) : Vehicle (b, m){
 void display Car () {
  display vehicle();
  cout <<"Type:" <<type <<end;
 }
};
 class ElectricCar : public Car{
private:
 int battery capacity;
Electric Car ~~public~~ ~~car~~ (string b, string m, string t,
 int battery): Car 1 (b, m, t)
{
 batteryCapacity = battery;
}
 void display Electric Car(){
  display Car();
  Cout <<"Battery :" << battery Capacity << "kwh"
   <<endl;
 }
};
int main ()
{
```

```
string brand, model, type;
int battery;
Cout << "Enter model:";
cin >> model;
cout << "Enter battery capacity:";
cin >> type;
cout << "Enter type:";
cin >> battery;
Electriclar e1 (brand, model, type, battery);
cout << "\n Electric Car Details:" << endl;
e1.display Electric Car ();
return 0;
}
```

output:-:
Enter brand : BMW
Enter model : 3
Enter type : SUV
Enter battery capacity : 100
Electric Car Details:
Brand = BMW
Model = 3
Type = SUV
Battery Capacity : 100.

iv) Hierarchial Inheritance.
Create a base class Employee with attributes emp ID and name. Derive two classes Manager and Developer from Employee. Manager has an attribute department and developer has an attribute programing language. Write a function to display details for both.

```
#Include <iostream>
using namespace std;
class Employe
{
  protected:
int emp ID;
 String name;
public:
Emblayee (int id, string n){
emp ID = id;
name = n;
}
  void display Employee ()
  cout << "Employee ID:" << empID << endl;
  cout << "Name:" << name << endl;
}
};
class Manager : public Employee {
private:
  String department;
Public:
Manager (int id, string n, string dept):
  Employee (id, n){
department = dept;
}
```

```
void display Manager(){
 display Employee();
 cout << " Department :" << department << endl;
}
};
class developer : Public Employee {
private:
 string programming language;
public:
 Developer (int id, string n, string lang):Employee
        (id, n){
 programming Language = lang;
}
  void display Developer(){
 display Employee();
 cout << " Programming language :" << programming
  Language << endl
 }
};
 int main()
{
 int id;
 string name, dept, lang;
 cout << " Enter Manager ID :";
 cin >> id;
 cout << " Enter Department :";
 cin >> dept;
 Manager m1 (id ; ~~name~~ dept);
 Cout << " \n Manager Details : \n";
 m1. display Manager();
 cout << " Enter Developer ID :";
 cin >> id;
 Cout << " Enter Developer Name :";
 cin >> name;
 cout << " Enter programming language :";
 cin >> lang;
 Developer d1 (id, name, lang);
 Cout << " \n Developer Details : \n";
 d1 . display Developer();
 return 0;
}
```

output:-

```
Enter Manager Id : 101
Enter Manager name : Durwanki
Enter Department : IT
Manager Details :
Employee ID : 101
Name : Durwanki
Department : IT
Enter Department ID : 102
Enter Developer Name : Durwa
Enter Programming language : C
Developer Details :
Employee ID : 102
Name : Durwa~~nki~~
Programming Language : C
```

iv) Hybrid Inheritance.
Combine multilevel and multiple Inheritance. Create a base class person with attributes name and age. Derive class student from Person. Create two classes sports and academics. Derive class result from student and sports. Write functions to calculate and display total marks and sports score along with student details.

```
#include <iostream>
using namespace std;
class person {
Protected:
string name;
int age;
public:
person (string n, int a) {
name=n;
age =a;
}
void displayPerson () {
cout <<"Name: " <<name <<endl;
cout <<"Age:" <<age <<endl;
}
};
class student : public person {
protected:
int rollNo;
public:
Student (string n, int a, int r): Person:(n,a) {
rollno=r;
}
void display Student () {
display Person ();
cout <<"Roll no.:"<<rollNo <<endl;
}
};
class Academics {
protected:
int marks;
public:
Academics (int m) {
marks =m;
}
};
class sports {
protected:
int score;
public:
Sports (int s) {
score =s;
}
};
class Result : public Student, public Academics,
public Sports {
public:
Result (string n, int a, int r, int m, int s):
student (n,a,r), Academics (m), sports (s) { }
void display result ();
displayResult ();
int total;
total = marks + score;
cout <<"Academics Marks:"<< marks <<endl;
cout <<"Sports Score:"<<Score<<endl;
cout <<"Total:"<<total<<endl;
```

```
}
};
int main (){
    string name;
int age, roll, marks, score;
cout <<" Enter name:";
cin>> name;
cout <<"Enter Age:";
cin>>age;
cout<<"Enter roll no:";
cin>>roll;
cout<<"Enter Academic Marks:";
cin >> marks;
cout<<"Enter sports score:";
cin >> "Student Result: \n", score;
cout<<" Student Result: \n";
Result r1 (name, age, roll, marks, score);
r1.displayResult ();
return 0;
}
```

output:

```
Enter name: Durwanki
Enter age :20
Enter Rollno. : 63
Enter Academic marks: 90
Enter sports score: 95
Student Result:
Name: Durwanki
Roll no: 63
Academic marks: 90
Sport score: 95
Total : 185
```

Experiment-7.

1) WAP using function overloading to calculating area of laboratory (rectangle) and class room (square).

```
#include <iostream>
using namespace std;
class area {
public:
 void calc (float len, float breadth){
 cout <<"Area of laboratory:" <<len * breadth
    <<endl;
}
 void calc (float side) {
 cout<<"Area of classroom:" << side * side<<endl;
};
int main() {
area a;
cout<<"Laboratory:"<<endl;
 a. calc (12.5, 10.0);
 cout <<"Classroom:" <<endl;
 a. calc (8.0);
 return 0;
}
```

2) WAP using function overloading to calculate sum of 5 float values and sum of 10 integer values.

```
#include <iostream>
using namespace std;
class sum{
public:
float sum(float a, float b, float c, float d,
    float e);
{ return a+b+c+d+e;}
int sum (int arr[], int n) {
int s=0;
for(int i=0; i<n ; i++)
  s+ =arr[i];
return s;
}};
int main () {
sum s;
cout <<" sum of 5 float no." <<s.sum
  (1.1, 2, 3, 4.4, 5.5)
int arr[10] = {1,2,3,4,5,6,7,8,9,10}
cout <<"Sum of 10 integers =" <<s.sum(arr,10);
return 0;
}
```

output:-

Sum of 5 float nos = 16.5
sum of 10 integers = 55.

3) WAP to implement unary operators when void with the object so that the numeric data member of class.

```
#include <iostream>
using namespace std;
class number {
int num;
public:
void setdata (int n) { num=n ; }
void display () { cout <<"value =" <<num<<
    endl ; }
void operator () {num = -num}
};
int main () {
number n;
n.setdata (10);
cout <<"Original :" ; n.display();
n;
cout <<"After Negation :" ; n.display();
return 0;
}
```

output:-
Original value = 10
After Negation value = -10.

4) WAP to implement the unary ++ operator.

```
#include <iostream>
using namespace std;
class counter{
int count;
public:
  void setdata(int c){count=c;}
  void disp(){cout<<"Count="<<count<<endl;}
  void operator++(int){count++;}
};
int main(){
counter C;
C.setdata(5);
cout<<"After pre-increment"; C.disp();
C++;
cout<<"Original"; C-disp();
C++;
cout<<"After post increment:";
return 0;
}
```

Output:-
Original: count=5
After Pre-increment: count=6
After Post-increment: count=7.

Experiment-8.

1) Operator overloading

```
#include <iostream>
using namespace std;
class Mystring{
string str;
public:
  void setstring(){
  cout<<" Enter string:";
  cin>>str;
  }
  void disp(){
  ~~cout<<"Enter string:";~~ cout<<str<<endl;
  ~~cin>>str;~~ }
  Mystring operator+(Mystring s){
  ~~void disp()~~ Mystring temp;
  temp.str=str+s.str;
  return temp;
};
int main(){
Mystring S1,S2,S3;
cout<<"First string";
S1.setString();
cout<<"Second string";
S2.setstring();
S3=S1+S2;
cout<<"Concatenated string:";
S3.disp()
return 0;
}
```

Output:-
First string:
Enter string: abc
Second string:
Enter string: xyz
concatenated string:
abc xyz

Page No.:
Date:
YOUVA

2) WAP to create a base class. Ilogin having data members names passwords. Declare accept() function virtual. Derive Email login membership log in classes from Ilogin. Display login details of employee.

```
#include <iostream>
using namespace std;
class Ilogin {
protected:
  string name, password;
public:
  virtual void disp(){
 cout<<"Name :"<<name<<"Password :"<<password;
 }
};
class EmailLogin : public Ilogin {
public:
  void accept() override{
 cout<<"\n Email login details";
 Ilogin ::accept()
 {
 void display () override{
 cout<<" Email login -> ";
 Ilogin :: display();
 }
};
class Membershiplogin :: public Ilogin {
public:
 void accept() override {
 cout<<"Membership login details";
 Ilogin :: accept();
}
```

```
class membershiplogin:: public Ilogin{
public:
void accept() override:{
cout << "Membership login Details";
Ilogin:: accept();
}
void display() override {
cout << "Membership login→";
Ilogin :: display();
}
};
int main(){
Email Login e;
Membershiplogin m;
e.accept();
m.accept();
cout << "Displaying Details:";
e.display();
m.display();
return 0;
}
```

output:-

```
Email login Details:
Enter password:alice123
Enter Name: Alice
Membership login Details:
Enter name : Bob
Enter password : bob123
```

Displaying login Details.
Email login → Name : Alice
password : alice 123
Membership login : Bob
password : bob123.

→ Experiment-9

1) WAP to copy contents of one file into another. Open "First.txt" in read (ios::in) mode and "second.txt" file in ~~write~~ write (ios::out) mode. copy elements of first into second. Assume "first" is already created.

```
#include <iostream>
using namespace std;
int main() {
ifstream fin ("First.text");
if (!fin) {cout << "First.text notfound";
return 0;}
string line;
while (getline (fin, line)){
fout << line << endl;
}
fin.close ();
fout.close ();
cout << " Context copied successfully";
return 0;
}
```

output :-
first.text not found.

2) WAP to count digits and spaces using file handling

```
#include <iostream>
using namespace std;
int main (){
ifstream fin ("First.txt");
if(!fin){ cout << "File not found!\n
"; return 0; }
char ch;
int digits=0; spaces=0;
while (fin.get(ch)){
 if (isdigit(ch)) digits++;
 if (ch==' ') spaces++;
}
 fin.close ();
cout<<" Digits:"<< digits <<" Spaces:"<<
spaces;
return 0;
```

output:-
File not found!

Exp-10
(By-mistake).

3) Calculator (10 operations)

```
# include <iostream>
using namespace std;
template (class T)
class calc (Ta,Tb){
public:
 calc (Ta,Tb){
  num1 = a;
  num2= b;
 }
  void display (){
 cout <<" Addition:"<<num1+num2;
 cout <<" Substraction:"<<num1-num2;
 cout<<"Multiplication:"<<num1*num2;
 cout<<" Division :"<<num1/num2;
 cout<<" Modulus :"<<num1%num2;
 cout <<" Square of num1:"<<num1*num1;
 cout<<" Square of num2 :"<<num2*num2;
 cout <<" Cube of num1 :"<<num1*num1*num1;
 cout <<" Cube of num2:"<<num2*num2*num2;
 cout <<" Avarage :" <<(num1+num2)/2;
 cout <<" Maximum :"<<((num1>num2)?
                num1 : num2);
 cout <<" Minimum "<<((num1<num2)?
                num1 : num2);
 }
};
```

```
int main (){
int a, b;
cout <<" Enter two integers:";
cin>>a>>b;
calc <int> calc (a,b);
cout <<" Calculate :"<<endl;
calc.disp ();
}
```

Output:-

```
Enter two integers : 10  5
Calculator
Addition : 15
Substraction : 5
Multiplication : 50
Division : 2
Modulus : 0
Square of num 1 : 100
Square of num 2 : 25
Cube of num 1 : 1000
Cube of num2 : 125
Average : 7.
Maximum : 10
Minimum : 5
```

3) WAP to count words using file handling.

```
#include <iostream>
# include <fstream>
# include <sstream>
using namespace std;
int main(){
ifstream fin("First.text");
if (!fin){
cout <<" File not found!";
return 0;
}
string word;
int count=0;
while (fin >>word){
count ++;
}
fin close ();
cout <<" Total words:" <<count <<endl;
return 0;
}
```

Output:-

File not found;

a) WAP to raccurence of a given word using file handling.

```
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
int main(){
ifstream fin("First.txt");
if (!fin){ cout<<" File not found!";
return 0;
}
string word, target;
int count=0;
cout<<"Enter words to search";
cin>> target;
while fin>> word {
if (word==target) ccount++;
}
fin.close();
cout<<" The word"<<target<<"occured
" << count <<" time";
return 0;
}
```

output:-
File not found!

Experiment-10

i) WAP to find sum of array elements using function template.

```
#include <iostream>
using namespace std;
template < class T>
T arrsum (T arr[]; int n){
T sum =0
for (int i=0; i<n ; i++)
sum + =arr[i];
return sum;
}
int main(){
int Arr[5] = { 1,2,3,4,5};
cout<< "Sum of array :"<<arr
sum (Arr 1,5);
}
```

output:

sum of array: 15

2) WAP of square function using template specialization calculate square of integer no(string). Write a specialized function string.

```
#include <iostream>
# include <string>
using namespace std;
template <class T>
T sq (Tn){
return n*n ;
}
template <>
String sq <string> astring) {
return sts;
}
int main (){
int i=5;
String str = "ABC"
cout<<"Square of Integer:"<< sqr( ;)<<endl;
cout <<"Square of String:" << sq(str)<<endl;
}
```

output:-
Square of Integer : 25
Square of String : ABCABC

3) Is done in 9th experiment.

d) WAP to implement push and pop methods from stack using class template

```
#include <iostream>
using namespace std;
template <class T>
class stack { Tarr [5] int top;
public:
 Stack() { top = -1 ;}
void push (Tval) {if (top ==4) cout<<"Stack
    overflow" ;
else
    arr [++top] = val;
}
 void pop () {if(top= = -1) cout<<"Stack
    underflow";
else
cout <<"Popped" <<arr [top = =] <<endl;}
void disp() {cout<<" Stack elements: ";
for (int i=0 ; i<=top ; i++)
 cout<<arr [i] << " ";
 cout <<endl;
}
};
 int main () {
 stack <int>5;
 S. push (10);
 S.push (20);
 S.push (30);
```

```
S.display();
S.pop();
S.display();
}
```

→ Experiment-11

1) WAP to create a vector, modify the value multiply by a scalar ~~vector~~ value. Display the vector.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
vector<int> v = {1,2,3,4,5,6,7,8,9,10};
cout<<"Initial vector :"<<endl;
for (int i=0;i<10;i++){
    cout<< v[i]<<" "<<endl;
}
cout <<"Multiply by 10" <<endl;
for (int i=0;i<10;i++){
  v[i] = v[i] * 10;
}
cout<<"New vector:" <<endl;
for (int i=0;i<10;i++){
  cout <<v[i]<<" " <<endl;
}
return 0;
}
```

→ Ep-11 (with iterators)

```
# include <iostream>
#include <vector>
using namespace std;
int main (){
vector<int> v={1,2,3,4,5,6,7,8,9,10};
cout<<"Initial vector:"<<endl;
for (vector<int>::iterator it=v.begin()
  ; it = v.end(); ++it ){
~~cout<<"Initial vector:"<<endl;~~
~~for (vector<int>::iterator it=v.be~~
  cout<<*it<<" ";
}
  cout<<endl;
cout<<"Multiply by 10"<<endl;
for (vector<int>::iterator it=v.begin();
it != v.end(); ++it){
*it = (*it) * 10;
}
cout<<"New vector:"<<endl;
for (vector<int>::iterator it = v.begin();
 it := v.end(); ++it){
cout<<* it<<" ";
}
cout<<endl;
return 0;
}
```

→ Experiment-12

1) WAP to implement Stack

```
# include <iostream>
#include <stack>
using namespace std;
stack<int> stack1;
void disp(){
stack<int>temp = stack1;
while(!temp.empty()){
cout<<temp.top()<<" ";
temp.pop();
}
cout<<endl;
}
int main(){
int num;
cout<<"Enter a no.:"<<endl;
cin>> num;
for (int i=0; i<num ;i++){
int temp;
cout<<"Enter element at position"<<i
  <<endl;
cin>>temp;
stack1.push(temp);
}
cout<<endl;
cout<<"Top most element:"<<stack.top()
  <<endl;
```

```
cout <<endl;
cout <<"Stack elements (top to
 bottom) ---> "<<endl";
disp()
cout <<endl;
cout <<"Pop Function:"<<endl;
 stack1.pop();
 disp();
stack1.pop();
 disp();
 stack1.pop();
 disp();
}
```

2) WAP to implement STL Queue.

```
#include <iostream>
# include <queue>
using namespace std;
int main(){
queue <int> q;
for (int i=0 ; i<=10 ; i++){
 q.push (i*10);
}
cout <<endl;
cout <<" front element" << q.front()<<
      endl;
cout<<" Back element"<<q.back()<<endl;
cout <<endl;
q.pop();
~~cout<<endl;~~
cout <<" After one pop, front="<<q.front()
   <<endl;
cout <<endl;
cout<<"Queue elements (front to back:)";
while (!q.empty()){
 cout <<q.front()<<" ";
 }
}
```