UNIT - 1

DATABASE SYSTEM CONCEPT

# OOP

## EXPERIMENT - 01

WAP in C++ language to declare a class student having data as rollno & name. Accept & display data for single student

```
#include <iostream.h>
using namespace std;
class Student
{
 int rollno;
 string name;
Public:
void accept ()
{
 cout << "Enter student name & roll no";
 cin >> name >> roll_no;
}
void display ()
{
cout << "Name of student=" << name;
```

```
cout << "Roll number of student :"
     << rollno;
}
int main()
{
Student S1;
S1.accept();
S1.display();
return 0;
}
```

WRITE A C++ CODE TO CREATE A CLASS BOOK HAVING DATA MEMBERS AS bname, bprice, bpages. Accept the data for two books & display the name of the book having greater price.

```
#include<iostream.h>
using namespace std;
class classbook
{ public:
  str bname;
  int bpages;
  int bprice;
  public:
 void accept()
{
cout<<"Enter the name of the book & the price of the and pages";
cin>>bname>>bpages>>bprice
}
void display()
{
 cout<<"Name of the book :"<<bname;
 cout<<"Price of the book :"<<bprice;
}
};
```

```
int main ()
{
book b1, b2;
 b1.accept ();
 b2.accept ();
if ( b1.price > b2.price )
 {
    b1.display ();
 }
else
{
   b2.display ();
}
}
```

Output :-

Enter the name, id, price of the book A 459 1234

Enter the name id, price of the book B 500 1245

Name of the Book : b
Price of the Book : 500
Id of the Book : 1245

3) WAP to declare a class "time" having data accept time in HH:MM:SS format, convert it into total seconds & display them

```
#include <iostream>
using namespace std;
class Time
{
 int H, M, S;
public:
void accept ()
{
cout << "Enter time in hours minutes seconds :";
cin >> H >> M >> S;
}
void calculate ()
{
 H = H * 3600;
 M = M * 60;
 S = S + H + M;
}
void display()
{
cout << "Total time in seconds :" << S;
};
```

```
int main()
{
 Time T1;
 T1.accept();
 T1.calculate();
 T1.display();
 return 0;
}
```

Output:-

```
Enter time in hours minutes seconds:
                    2 46 37
Total time in seconds : 9997
```

29/7/25

## EXPERIMENT : 02

a) WAP to declare a class 'city' having data members as name & population. Accept this data for 5 cities having highest population.

```
#include <iostream>
using namespace std;
class city
{
string name;
 int pop;
public:
void accept()
{
 cout<<"enter city name & populatio
 cin>> name >> pop;
}
void display()
{
 cout<<"Name of city(having highest
 cout<<"Population:"<<name;
}
};
```

```
int main()
{
city c[5];
int i, max;
for(i=0; i<5; i++)
{
 c[i].accept();
}
max = c[0].pop;
for(i=0; i<5; i++)
{
 if(c[i].pop > max)
 {
  max = i;
 }
}
c[max].display();
return 0;
}
```

Output:

```
Enter City Name: A
Enter City Population: 01
Enter City Name: B
Enter City Population: 02
Enter City Name: C
Enter City Population: 03
Enter City Name: D
Enter City Population: 04
Enter City Name: D
Enter City Population: 05
City Having Highest Population: D
```

(2) WAP to declare a class 'Account' having data members as Account number & balance. Accept this data for 10 accounts & give Interest of 10% where balance is equal or greater than 5000 & display them.

```
#include<iostream>
using namespace std;
class Account
{
 int acc_no;
 float bal;
public:
 void accept()
 {
 cout<< "Enter Account Number:
 cin >> acc_no;
```

```
cout<<"Enter balance";
cin>>bal;
}
void display()
{
cout<<"Account number:"<<acc_no;
cout<<"Balance:"<<bal;
}
};
int main()
{
Account A[10];
int i;
for(i=0; i<10; i++)
{
A[i].accept();
}
for(i=0; i<10; i++)
{
if(A[i].bal>=5000)
A[i].bal = A[i].bal + (0.1*A[i].bal);
A[i].display();
}
}
return 0;
}
```

Output :-

```
Enter Account Number : 2564
Enter Account Balance : 2000
                      : 4879
                      : 7890
                      : 6789
                      : 4567
                      : 3578
                      : 2478
                      : 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 : 62960.8
```

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

```
#include <iostream>
using namespace std;
class staff
{
string name;
string post;
public:
void 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

24/7/25

## Experiment : 3

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

```
#include <iostream>
using namespace std;
class book
{
 string btitle;
 string author_name;
 int price;
 public:
 void accept ()
{
cout << "Enter book title";
cin >> btitle;
cout << "Enter author name";
cin >> authorname;
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: Kamya
Enter Price of Book: 2500
Book Title: Clglife
Author Name: Kamya
Book Price: 2500
```

6) WAP to declare a class 'student' having data members roll_no & percentage. Using 'this' pointer invoke member functions to accept & 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

Q

```
#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 number \n";
cin >> roll;
}
void display ()
{
cout << "The name of student" << name;
cout << "The roll no \n" << roll;
}
class marks
{
 public:
 int m1, m2;
void accept1()
{
```

```
cout << "Enter marks of two subject:";
cin >> m1 >> m2;
}
void display1()
{
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.accept1();
 s1.display();
 m.display1();
 return 0;
}
```

Output:

```
Enter the name: Kavya
Enter roll no: 72
Enter the name: Kavya
The roll no: 72
Enter two marks:
32
16
Percentage: 48
```

Qn 5 18/9

# EXPERIMENT - 4

1) 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 << "\nAfter 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
```

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

```
#include <iostream>
using namespace std;
class number{
  private:
    int 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
```

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

```
#include <iostream>
using namespace std;
class num2;
class num1 {
    private:
        int a;
    public:
        void accept() {
        cout << "Enter the first number: ";
        cin >> a;
    }
    friend void swap(num1 &num1, num2 &);
};
class num2 {
        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;
    num2 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
```

4) WAP to create two classes Result1 & Result2 which stores the marks of the students. Read the value of marks for both the class objects & compute the average of two results.

```
#include<iostream>
using namespace std;
class result2;
class result1 {
    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(result1, result2);
};
class result2 {
    private:
    float marks;
    public:
    void accept(){
    cout<<"Enter the total marks obtained in the second semester out of 100:";
    cin>>marks;
    }
    friend void average(result1, result2);
};
void average(result1 x, result2 y){
    float avg;
    avg = (x.marks + y.marks)/2;
    cout<<"Avg of both the results:"<<avg;
}
int main(){
    result1 r1;
    result2 r2;
    r1.accept();
    r2.accept();
    average(r1, r2);
    return 0;
}
```

Output :-
Enter student name : Kamya Chopra
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

⑤ WAP to find the greatest number among two numbers using friend function.

```
#include <iostream>
using namespace std;
class num2;
class num1 {
    private:
    int a;
    public:
    void accept()
{
cout << "Enter End number";
cin >> b;
}
friend void grt(num1, num2);
};
void grt(num1 x, num2 y)
{
  if (x.a > y.b)
{
cout << x.a << " is greater than" << y.b;
}
else {
cout << y.b << " is greater than" << x.a;
}
}
int main() {
num n1;
num2 n2;
n1.accept();
n2.accept();
grt(n1, n2);
return 0;
}
```

Output :-
Enter the first number : 8
Enter the second number : 10
10 is greater than 8

# FRIEND FUNCTION PRACTICE

1) Create 2 classes, Class A and Class B each with a private. Write a friend sum() that can access private data from both classes & return the sum.

```
#include <iostream>
using namespace std;
class B;
class A {
    private:
      int a;
    public:
  void accept() {
cout << "Enter the first number";
cin >> a;
}
  friend void sum (A, B);
};
class B {
    private:
      int b;
    public:
    void accept()
{ cout << "Enter the second number";
cin >> b;
}
  friend void sum (A, B);
};
void sum (A x, B y)
{
  int sum;
  sum = x.a + y.b;
  cout << "Sum:" << sum;
}
int main (){
  A n1;
  B n2;
  n1.accept ();
  n2.accept ();
  sum (n1, n2);
}
```

Output:

```
Enter the first number:5
Enter the second number:6
Sum = 11
```

Q WAP with a class Number that contains a private integer. Use a friend function swapNumbers (Number&, Number&) to swap the private values of two Number objects.

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

Output:

```
Enter the first number: 7
Enter the second number: 4

After swapping:
First number: 4
Second number: 7
```

3) Define 2 classes Box and Cube, each having a private volume. Write a friend function friend (Box Cube) that determines which object has a larger volume.

```
#include <iostream>
using namespace std;
class Box;
class Cube {
    private:
    float volume;
    public:
    void accept() {
cout<<"Enter the volume of a cube:";
cin >> volume;
}
friend void findgrt (Cube, Box);
};
class Box {
 private:
  float vol;
 public:
  void accept()
{
cout<<"Enter the volume of the Box:";
cin >> vol;
}
friend void findgrt (Cube, Box);
};
void findgrt (Cube x, Box y){
 if (x.volume > y.vol){
cout<<"Cube has larger volume";}
else {
cout<<"Box has larger volume";
}
}
int main(){
   Cube c;
   Box b;
   c.accept();
   b.accept();
   findgrt(c,b);
   return 0;
}
```

Output:

```
Enter the volume of a cube : 567.88
Enter the volume of a Box : 985.66
```

Box has larger volume.

Date: YOUVA

④ Create a class Complex with real & imaginary parts as private members. Use a friend function to add two complex numbers & return the result as a new complex object.

```
#include<iostream>
using namespace std;
class Complex {
private:
 int r, i;
public:
 void accept() {
 cout<<"Enter real part:";
 cin>>r;
 cout<<"Enter imaginary part:";
 cin>>i;
 }
 void display(){
  cout<<r<<"+"<<i<<"i"<<endl;
 }
 friend Complex sum(Complex, Complex);
};
Complex sum(Complex x, Complex y){
      Complex temp;
      temp.r = x.r + y.r;
      temp.i = x.i + y.i;
      return temp;
}
int main(){
 Complex c1, c2, c3;
 cout<<"Enter the first complex number";
  c1.accept();
 cout<<"Enter the second complex number";
  c2.accept();
 cout<<"Complex Number 1";
  c1.display();
 cout<<"Complex Number 2";
  c2.display();
 c3 = sum(c1, c2);
 cout<<"Sum of 2 complex numbers=";
 c3.display();
 return 0;
}
```

Output:

```
Enter the first complex number:
Enter the real part:5
Enter the imaginary part:6
Enter the second complex number:
Enter the real part:8
Enter the imaginary part:2
Complex Number 1:
5+6i
```

Complex Number 2:

8 + 2i

Sum of the two complex numbers = 13+8i

(5) Create a class Student with private data members: name and three subject marks. Write a friend function calculate Average (Student) that calculates & displays avg marks.

```
#include <iostream>
using namespace std;
class Student{
    string name;
    int math;
    int phy;
    int chem;
    public:
    void accept(){
cout<<"Enter student name:";
cin>>name;
cout<<"Enter maths marks";
cin>>math;
cout<<"Enter physics marks";
cin>>phy;
cout<<"Enter chem marks:";
cin>>chem;
}
friend void calculateAverage(Student);
};
void calculateAverage(Student x)
{
  float avg;
  avg = (x.math + x.phy + x.chem)/3;
  cout<<"Avg marks="<<avg;
}
int main(){
Student s;
s.accept();
CalculateAverage(s)
return 0;
}
```

Output:

```
Enter student name: Kamya
Enter maths marks: 98
Enter physics marks: 96
Enter chem marks: 92

Avg marks = 95
```

6) Create 3 classes: Alpha, Beta & Gamma each with a private data member. Write a single friend function that can access all three and print their sum.

```
#include<iostream>
using namespace std;
class Beta;
class Gamma;
class Alpha {
    int a;
  public:
  void accept(){
cout<<"Enter 1st number:";
cin>>a;
}
friend void sum(Alpha, Beta, Gamma);
};
class Beta {
    int b;
  public:
void accept(){
cout<<"Enter 2nd number:";
cin>>b;
}
friend void sum(Alpha, Beta, Gamma);
};
class Gamma {
    int g;
  public:
    void accept(){
cout<<"Enter 3rd number";
cin>>g;
}
friend void sum(Alpha, Beta, Gamma);
};
void sum(Alpha x, Beta y, Gamma z)
{
  int sum;
  sum = x.a + y.b + z.g;
  cout<<"sum:"<<sum;
}
int main()
{
  Alpha a;
  Beta b;
  Gamma g;
  a.accept();
  b.accept();
  g.accept();
  sum(a, b, g);
  return 0;
}
```

Output:
Enter 1st number : 6
Enter 2nd number : 3
Enter 3rd number : 9

Sum = 18

⑦ Create a class point with private members x and y. Write a friend function that calculate & returns the distance between two point objects.

```
#include <iostream>
#include <math>
using namespace std;
class Point {
        int x, y;
        public:
void accept () {
cout << "Enter x coordinate";
cin >> x;
cout << "Enter y coordinate";
cin >> y;
}
void display () {
cout << "(" << x << "," << y << ")";
}
friend double diff (Point, Point);
};
double diff (Point x, Point y)
{
   double temp;
return temp = sqrt (power (x.x - y.x) +
point ((x.y - y.y), 2));
}
int main () {
 Point p1, p2;
cout << "For the point P";
 p1.accept ();
cout << "For the point Q";
 p2.accept ();
cout << "P =";
p1.display ();
cout << "Q =";
p2.display ();
cout << "Diff btw the two points =
     << diff (p1, p2) << "units";
}
```

Output :-
For the Point P:
Enter x co-ordinates : 5
Enter y co-ordintes : 6
For the Point Q:
Enter x co-ordinate : 7
Enter y co-ordinate : 8
P : (5, 6)
Q : (7, 8)

Diff btw the two point = 2.82843 units

8) Create two classes : BankAccount and Audit BankAccount holds private balance info. Write a friend function in Audit that accesses & points balance info for auditing.

```
#include <iostream>
using namespace std;
class Audit;
class BankAccount {
    private:
        float balance;
    public:
    void accept () {
        cout << "Enter the balance info";
        cin >> balance;
    }
    friend class Audit;
};
class Audit {
    public:
    void display (BankAccount x)
    {
    cout << "Balance Information =" <<
                                x.balance;
    }
};
int main () {
    BankAccount b;
    Audit a;
    b.accept ();
    a.display (b);
    return 0;
}
```

Output :
Enter the balance information : 9867.49
Balance information : 9867.49

9/9/25

# EXPERIMENT - 5

a) (i) WAP 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, i, sum = 0;
public:
Sum ()
{
   n = 20;
}
void calculate ()
{
 for (i = 1; i <= n; i++)
 {
    sum = sum + i;
 }
}
void display ()
{
cout << "Sum = " << sum;
};
```

```
int main ()
{
 Sum On;
 n.calculate();
 n.display();
return 0;
}
```

Output:
Sum = 210

(ii) (a) Using parameterised constructor.

```
#include <iostream>
using namespace std;
class Sum
{
 int n, i, sum = 0;
 public:
  Sum (int num)
{
   n = num;
}
 void calculate ()
{
  for (i = 01; i <= n; i++)
```

```
{
  sum = sum + i;
}
}
void display ()
{
cout << "Sum=" << sum;
}
};
int main ()
{
Sum n (20);
n.calculate;
n.display;
return 0;
}
```

Output:
Sum = 210

(6) WAP to declare a class "student" having data members as name and percentage. Write a constructor to initialize the data members. Accept & display data for one student.

```
#include <iostream>
using namespace std;
class Student
{
 int per;
 string name;
 public:
 Student ()
 {
   name = "Kavya";
   per = "98";
 }
 void display ()
 {
 cout << "Name:" << name;
 cout << "Percentage:" << per;
 }
 };
 int main () {
```

```
Student s;
s.display();
return 0;
}
```

Output:
Name: Kamya
Percentage: 98

(ii) Parametrized Constructor

```
#include <iostream>
using namespace std;
Student Cname
{
 int per;
 string name;
public:
 Student (string n, int percent
{
  name = n;
  per = percent;
}
void display (){
cout << "Name:" << name;
cout << "Percentage:" << per;
}
};
```

```
int main()
{
Student s("Kamya", 96)
s.display();
return 0;
}
```

Output:
Name = Kamya
Percentage: 96.

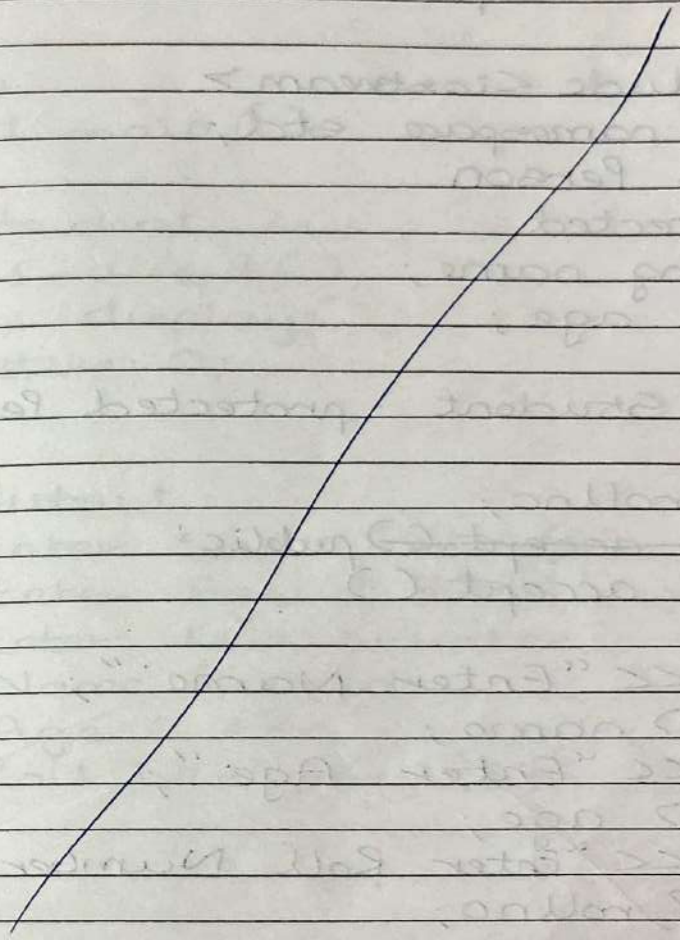
(c)
```
#include <iostream>
using namespace std;
class College {
int roll_no;
string name, course;
public:
College()
{
name = "Kamya";
rollno = 71
course = "Computer engineering";
}
College(string a, int b, string c = "
                                Computer
{
```

```
name = a;
rollno = b;
course = c;
}
void display()
{
cout << "In Name is : " << name;
cout << "In Rollno is : " << roll_no;
cout << "In Course is : " << course;
}
};
int main()
{
college c("Mufaddal", 35), ce("Kanya
                                71);
c.display();
return 0;
}
```

(d)

```
#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, city)
{
   name = "rahul";
   nof = x;
   nos = y;
}
void display() {
```

```
cout << " In Name is :" << n;
cout << "In Number of faculty is :" << nof;
cout << " In Number of number is : " << nos;
 }
};
int main ()
{
 department d1();
 d2 ("William", 80, 60);
 d3 (80, 75);
 d1. display ();
 d2. display ();
 d3. display ();
 return 0;
}
```



# Experiment - 06

① 
```
#include <iostream>
using namespace std;
class Person
{protected
 string name;
 int age;
};
class Student : protected Person
{
 int rollno;
 void accept() public:
 void accept()
{
cout<<"Enter Name:";
cin>>name;
cout<<"Enter Age:";
cin>>age;
cout<<"Enter Roll Number:";
cin>>rollno;
}
void display()
{
cout<<"Name:"<<name<<endl;
cout<<"Age:"<<age<<endl;
cout<<"Roll Number:"<<rollno;
}
};
int main()
{
student s1;
s1.accept();
s1.display();
return 0;
}
```

Output:

```
Enter Name: Kanya
Enter Age: 17
Enter Roll Number : 71
Name: Kanya
Age : 17
Roll Number : 71
```

(2)

```
#include <iostream>
using namespace std;
class Academics
{ protected:
 int marks;
};
class Sports
{
protected:
int ss;
};
class Result : public Academics, public Sports
{
 public:
int ts;
void accept()
{
cout << "Enter marks:";
cin >> marks;
cout << "Enter sports score:";
 cin >> ss;
}
void calculate() {
ts = marks + ss;
cout << "Total score:" << ts;
}
void display() int main()
{
result r;
r.accept();
r.calculate();
return 0;
}
```

O/P

```
Enter marks: 45
Enter sports score: 30
Total score: 75
```

③
```
#include <iostream>
using namespace std;
class Vehicle
{
public:
string branch;
string model;
};
class car : public Vehicle {
  string attribute;
};
class ElectricCar : public Car {
varchar
int batteryCapacity;
public:
void accept()
{
cout<<"Enter branch:";
cin >> branch;
cout<<"Enter model:";
cin >> model;
cout<<"Enter attribute:";
cin >> attribute;
cout<<"Enter batterycapacity:";
cin >> batteryCapacity;
}
void display()
{
cout<<"Branch:"<<branch;
cout<<"Model:"<<model;
cout<<"Attribute:"<<attribute;
cout<<"Battery Capacity:"<<batterycapacity;
};
int
void main()
{
ElectricCar c1;
c1.accept();
c1.display();
return 0;
}
```

O/P

```
Enter branch: Suzuki
Enter model : Land cruiser
Enter Attribute: SUV
Enter batterycapacity: Not Electric
Branch: Suzuki
Model: Land cruiser
Attribute: SUV
batterycapacity: Not Electric
```

④
```
#include <iostream>
using namespace std;
class Employee
{public:
int empid;
string name
};
class Manager : public Employee
{
string department;
public:
void accept()
{
cout << "Enter employee id";
cin >> empid;
cout << "Enter name";
cin >> name;
cout << "Enter department name";
cin >> department;
}
void display()
{
cout << "Employee id" << empid;
cout << "Name:" << name;
cout << "Department:" << department;
}
};
```

```
class Developer : public Employee
{
string programminglanguage;
public:
void accept()
{
cout << "Enter employee id";
cin >> empid;
cout << "Enter name";
cin >> name;
cout << "Enter programming language";
cin >> programminglanguage;
}
void display()
{
cout << "Employee id:" << empid;
cout << "Name:" << name;
cout << "Programming Language:" << programminglanguage;
}
};
int main()
{
Manager m;
m.accept();
```

m.display();
Developer d;
d.accept();
d.display();
return 0;
}

Output:
Enter employee id: 101
Enter name: A
Enter department name: CSE
Employee id: 101
Name: A
Department: CSE
Enter employee id: 201
Enter name: B
Enter Programming Language: Java
Employee id: 201
Name: B
Programming Language: Java

(5)
```
#include <iostream>
using namespace std;
class Person
{
 protected:
 string name;
 int age;
};
class Student : protected Person
{
 protected:
 void accept()
 {
  cout<<"Name of Person:";
  cin>>name;
  cout<<"Age of Person:";
  cin>>age;
 }
 void display()
 {
 cout<<"Name:"<<name;
 cout<<"Age:"<<age;
 }
};
class Sports
{ protected:
 int sportsscore;
}
```

```
class Academics
{
protected:
 int academicscore;
};
class Result: protected sports,
                    academics
{
protected:
 int total;
void accept()
{
 cout<<"Enter sports score:";
 cin>>sportsscore;
 cout<<"Enter academic score:";
 cin>>academicscore;
}
 void display()
{
cout<<"Sports score:"<<sportsscore;
cout<<"Academic score:"<<academicscore;
void calculate()
{
 total=sportsscore+academicscore;
}
void display()
{
cout<<"Total="<<total;
}
};
int main(){
Student s;
s.accept();
s.display();
Result r;
r.accept();
r.calculate();
r.display();
return 0;
}
```

Output:-

```
Name of Person: A
Age of Person: 13
Name: A
Age: 13
Enter sports score: 45
Enter academic score: 45
total = 45 + 45
Total = 90
```

⑥

```
#include <iostream>
using namespace std;
class CollegeStudent {
protected:
  int stu_id;
  string ccode;
public:
  void accept() {
cout << "Enter Student Id";
  cin >> stu_id;
cout << "Enter college code:";
  cin >> ccode;
}
void display() {
cout << "Student id:" << stu_id;
cout << "College Code:" << ccode;
  }
};
class Test : virtual public CollegeStudent
{
  protected:
    float percentage;
  public:
    void accept() {
CollegeStudent :: accept();
cout << "Enter Test percentage:";
cin >> percentage;
}
void display() {
CollegeStudent :: display();
cout << "Test Percentage:" << percentage;
}
};
class Sports : virtual public CollegeStudent
{
  protected:
    char grade;
  public:
    void accept() {
cout << "Enter sports grade:";
  cin >> grade;
}
void display() {
cout << "Sports Grade:" << grade;
}
};
class Result : public Test, public Sports {
    float total_marks;
  public:
    void accept() {
      CollegeStudent :: accept();
```

```
cout << "Enter test percentage";
cin >> percentage;
cout << "Enter sports grade";
cin >> grade;
cout << "Enter Total marks";
cin >> total_marks;
}
void display(){
 CollegeStudent :: display();
cout << "Test Percentage: " << percentage;
cout << "Sports Grade: " << grade;
cout << "Total Marks: " << total_marks;
}
};
int main(){
 Result r;
cout << "Enter Student Details";
 r.accept();
cout << "Student Result";
 r.display();
 return 0;
}
```

Output :-

```
Enter Student Id: 101
Enter College Code: 1262241655
Enter test percentage: 50
Enter sports grade: A
Enter Total marks: 90
Student Id: 101
College Code: 1262241655
Test Percentage: 50
Sports Grade: A
Total Marks: 90
```

# Experiment : 7

Q

```
#include <iostream>
using namespace std;
class Area {
public:
float calculate (float length, float breadth)
{
  return length * breadth;
}
float calculate (float side) {
  return side * side;
}
};
int main () {
Area a;
cout<<"Area of laboratory :"<<a.calculate(10,6);
cout<<"Area of square:"<<a.calculate(5);
}
```

Output:

```
Area of laboratory: 60
Area of square: 25
```

Q

```
#include <iostream>
using namespace std;
class Sum {
public:
 int total (int a[], int n){
   int s = 0;
for (int i = 0; i<n; i++){
    s+ = a[i];
    return s;
}
float total (float a[]; int n){
     float s = 0;
for (int i = 0; i<n; i++){
      s+ = a[i];
      return s;
  }
};
int main (){
  Sum s;
int marks[5] = {1, 2, 3, 4, 5};
float grades[5] = {5.0, 6.0, 7.0, 8.0, 9.0};
cout<<"Sum of 10 student marks:"
    <<s.total(marks, 5)<<endl;
cout<<"Sum of 5 student grade:"
    <<s.total(grade, 5)<<endl;
return 0;
}
```

Output:
Sum of 105 student marks: 15
Sum of 5 student grade points: 25

Q3)

```
#include <iostream>
using namespace std;
class Teacher {
    int experience;
public:
    Teacher (int e) {
        experience = e;
    }
void display() {
cout << "Experience :" << experience << "year" << endl;

}
void operator - () {
    experience = - experience;
    }
};
int main () {
    Teacher t1(10);
    t1.display();
    -t1;
```

```
cout << "After negation:";
    t1.display()
    return 0;
}
```

Output

Q4)

```
#include <iostream>
using namespace std;
class Student {
    int count;
public:
    Student (int c = 0) {
        count = c;
    }
void operator ++ () {
        ++count;
    }
void operator ++ (int) {
    count ++;
    }
void display () {
    cout << "Student count:" << count;
    }
};
```

```
int main () {
Student s1 (50);
cout << "Before increment :" << endl;
s1 display ();
++ s1;
cout << "After pre-increment :" << endl;
s1.display ();
s1 ++;
cout << "After post increment :" << endl;
s1.display ();
return 0;
}
```

Output:

50
51
51

Experiment : 8.

8)

```
#include <iostream>
#include <string>
using namespace std;
class Combine {
    string str;
public:
    Combine (string s = "") {
        str = s;
    }
Combine operator+ (Combine &obj) {
 return Combine (str + obj.str);
 }
void display () {
 cout << str << endl; }
};
int main () {
Combine s1("xyz"), s2("per"), s3;
   s3 = s1 + s2;
cout << "Concatenated string:";
 s3.display ();
}
```

Output:

xyzper

Q) #include <iostream>
#include <string>
using namespace std;
class I login {
protected:
string name, password;
public:
virtual void accept () {
cout << "Enter name:";
cin >> name;
cout << "Enter password:";
cin >> password;
}
virtual void display () {
cout << "Name:" << name;
cout << "Password:" << password;
}
};
class Email login : public I login {
string email;
public:
void accept () override {
cout << "Enter Email ID:";
cin >> email;
login :: accept ();
}
void display () override {
cout << "\n ---Email Login Details---";
cout << "Email Id:" << email << endl;
I login :: display ();
}
};
class Membership login : public I login
{
string member ID;
public:
void accept () override {
cout << "Enter Membership ID:";
cin >> member ID;
I login :: accept ();
}
void display () override {
cout "\n ---Membership login details
---" << endl;
cout << "\n Membership ID :" <<
membership ID;
I login :: display ();
}
};
int main () {
I login login;
Email login;

```
MembershipLogin m;
login: se;
login → accept();
login → display();
login = sm;
login → accept();
login → display();
return 0;
}
```

Output:
Email Login Details
Enter Name: Alice
Enter password: alice123

Membership login Details
Enter name: Bob
Enter password: bob123

Displaying login Details
Email login → Name: Alice
password: alice123
Membership login → Name: Bob
password: bob123

## EXPERIMENT - 9

(a)
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
ifstream fin("First.txt");
ofstream fout("Second.txt");
if (fin) {
cout << "First.txt not found";
return 0; }
string line;
while (getline(fin, line)) {
fout << line << endl;
}
fin.close();
fout.close();
cout << "Contents copied successfully";
return 0;
}
```

Output:
first.txt not found

⑥

```
#include <iostream>
#include <fstream>
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!

⑥

```
#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;
int count = 0;
while (fin >> word) {
 count++;
}
fin.close();
cout << "Total words: " << count;
return 0;
}
```

Output:-
File not found!

```
#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 word to search";
cin >> target;
while fin >> word {
if (word == target)
  count++;
}
fin.close();
cout<< "The word"<< target<< "Occured
<<count<< "time";
return 0; }
```

Output:-
File not found!

## Experiment 10

```
#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 t = arr[i];
    return sum;
  }
}
int main () {
int Arr1[5] = {1, 2, 3, 4, 5};
cout<< "Sum of Array:"<< arr sum (Arr1, 5);
}
```

Output:

Sum of array: 15

(b)
```
#include <iostream>
#include <string>
using namespace std;
template <class T> void fun(T a)
{
  T m = 0;
  m = a*a;
cout<< "The square of integer is:"<<
                          m<<endl;
}
template <> void fun (string str)
{
  string m;
  m = str + str;
cout<< "The concatinated string is:"
                          <<m<<endl;
}
int main()
{
fun<2> int n;
fun cout<< "Enter a value of n";
  cin >>n;
string s;
cout<<"Enter values of s;
  cin >> s;
fun(n);
fun(s);
return 0;
}
```

```
Enter a value of n 5
Enter a value of s KD
The square of a integer is: 25
The concatinated string is: KDKD
```

(c)
```
#include <iostream>
using namespace std;
template <class T>
class calc {
  T num1, num2;
public:
calc (Ta, T6){
public:
calc (Ta, T6){
  num1 = a;
  num2 = b;
}
void display (){
cout<< "Addition: " << num1 + num2;
cout<< "Subtraction: "<< num1 - num2;
cout<< "Multiplication: "<< num1 * num2;
cout<< "Division: "<< num1 / num2;
cout<< "Modulus: "<< num1 % num2;
cout<< "Square of num1: "<< num1 * num1;
cout<< "Cube of num1: "<< num1*num1*num1;
cout<< "Square of num2: "<< num2 * num2;
```

```
cout<< "Cube of num2:" <<num2* num2* num2;
cout<< "Average:" << (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.display();
}
```

Output:

Enter two integers: 10 5
Calculator
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
Modulus: 0
Square of num1: 100
Cube of num1: 1000
Square of num2: 25
Cube of num2: 125
Average: 7
Maximum: 10
Minimum: 5

(d)

```
#include <iostream>
using namespace std;
template <class T>
class stack{
 T arr[5];
 int top;
public:
Stack()
{
 top = -1;
}
void push(T value)
{
 if (top == 4)
 cout<< "Stack Overflow";
else
 arr[++top] = value;
}
void pop(){
 if (top == -1){
 cout<< "Stack underflow";
}
else
cout<< "Popped" << arr[top--] <<endl;
}
void display(){
```

```
cout << "Stack elements";
for (int i = 0; i <= top; i++)
{
    cout << arr[i] << " " << endl;
}
};
int main()
{
stack <int> s;
s.push(10);
s.push(20);
s.push(30);
s.display();
s.pop();
s.display();
}
```

Experiment - 11

```
#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;
}
```

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

11/4

## Experiment - 12

```
#include <iostream>
#include <stack>
using namespace std;
stack<int> stack1;
void display(){
 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+1;
cin>>temp;
stack1.push(temp);
}
cout<<endl;
```

```
cout << "Top most element: " << stack
                                  top();
cout << endl;
cout << "Top most element: " <<
                                  stack.top();
cout << endl;
cout << "Stack elements (top to bottom): ";
display();
cout << endl;
cout << "Pop Function: " << endl;
stack1.pop();
display();
stack1.pop();
display();
stack1.pop();
display();
}
```

Q

```
#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();
cout << "Back element " << a.back();
cout << endl;
q.pop();
cout << "After one pop, front = " << q.
                                  front();
cout << endl;
cout << "Queue elements (front to back): ";
while(!q.empty()){
 cout << q.front() << " ";
}
}
```