Experiment : 1

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

Code :

```
# include <iostream.>
using namespace std;
Class student
{
int roll_no;
String name;
public :
void accept ()
{
Cout << "Enter student name and roll no :";
Cin >> name >> roll-no;
}
void disp ()
{
Cout << "Name of student : " << name;
Cout << " In Rollno. : " << roll no;
};
};
int main ()
{
Student s1;
s1 . accept ();
s1 disp ();
```

```
return 0;
}
```

2) Write a C++ code to create a class book having data members as Bname, Bprice, Bpapers. Accept the data for 2 Books & display the name of the book having greater price.

Code:

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

class Classbook
{
Public:
int Bprice, Bpages;
string Bname;
Public:
void accept ()
{
Cout << "Enter book name, price and pages ;
Cin >> Bname >> Bprice >> Bpages ;
}
void disp ()
{
Cout << "The name of the most expensive book
is "Bname ;
}
```

3)

```
int main ()
{
 classbook b1, b2 ;

b1.accept ();
b2. accept ();

if( B1 . Bprice > B2 . price >
{
 b1. disp();
}
else
{
b2. disp ()
}
return 0;
}
```

3) Write a program to declare class "time", accept time in HH:MM:SS format, convert it into seconds and display them.

Code:

```
# include <iostream>
using namespace std;
class time
{
Public:
int hour min sec:
Void accept ()
{
Cout << "Enter time in hours, minutes and
seconds"
Cin >>" hour >> min >>sec;
}
Void accept disp ()
{

int S1 = (hour * 3600) + (min * 60) + sec;
Cout << "The time in second " << S1;
}
};
int main ()
{
class time + 1;

t1. accept ();
t2. disp();
}
```

5/8/25

Experiment - 2

1] WAP to declare class 'city' having data member as name and population. Accept this data for 5 cities having highest population.

```
# include <iostream>
using namespace std;
class city
{ public:
string name;
int population;
public:
void accept ()
{
Cout << "Enter the name of city: " << endl;
Cin >> name;
Cout << "enter population: " << endl;
Cin >> population;
}
void display ()
{
Cout << "Flame: " << name << endl;
Cout << "Population: " << population << endl;
}
};
city C(5);
int main ()
{
int i;
for (i=0; i<5; i++)
{
```

```
c[i].accept()
}
int max = 0;
for(i=0; i<5; i++)
{
if(c[i].population > c[max].population)
{
  max = i;
  }
}
cout<<"The city with maximum population:"<<
endl;
c[max].display();

return 0;
}
```

Output ....

```
Enter name of city:
City 1: Pune
Population: 234572
City 2: Katraj
Population: 234571
City 3: Amritsar
Population: 2345
City 4: Mahabaleshwar
population: 1234
City 5: Vai
Population: 2345678

The City with maximum population: 2345678
```

2]

```
#include <iostream>
using namespace std;
class account
{
  int acc no;
  int balance;

public:
   void accept ()
{
cout <<"Enter account number : "<< endl;
cin >>" acc no;
cout <<"Enter balance : "<<end l ;
 cin >> balance;
}
 void display ()
{
cout <<" Account numbers: "<<accno<<"
    Balance : "<< balance <<end l;

}
  void add intrest []
{
  if (balance >= 5000]
{
  int intrest = balance * 0.10;
   balance += intrest;
cout <<" Intrest of 10% added," << balance
    << end l;
}
else
```

```
{
Cout << " Balance is less than 5000, no intrest
is added. " << end l;
}
}
int get Balance ()
{
return balance;
}
};

int main ()
{
account accounts [10];
int num Accounts;

Cout << "Enter number of accounts (max 10):";
Cin >> num Accounts;

for (int i=0; i<num Accounts ; i++)
{
Cout ("In Account " << (i+1) << ": " << end l;
accept accounts [i]. aceept ();
}
Cout << "In Account Display with 10% intrest " <<
end l;
for (int i=0; i<num Accounts ; i++)
{
Cout << "In Account " << (i+1) << ": " << end l;
account [i]. display ();
account [i]. addintrest ();
}
```

```
return 0;
}
```

3)
```
#include <iostream>
using namespace std;
class staff
{
 string name;
 string post;
public:
 void accept ()
{
 Cout <<"Enter name:"*;
  Cin >> name;
 Cout <<"Enter post:" ;
 Cin >> past;
}
 void display ()
{
Cout <<" Name :"<<name<<,"Post :"<< post <<
 end l;
}
string get Post ()
{
return post;
}
string get name ()
 {
 return name;
 }
```

```
int main
{
staff s[
bool foun

for (int i
{
Cout <<"
s[i].ac
}
for (int
{
if (s[i
{
cout <<
found +

break;
}
}
if (!found
Cout <<"

return
}

Output

Name: A
Post: St
```

```
int main ()
{
Staff S[5] ;
bool found HOD = False;

for (int i = 0; i<5 ; i++)
{
Cout <<" Staff member " <<(i+1)<<" : " <<end l;
S[i). accept ():
}
 for (int i = 0; i<5; i++)
{
 if (S[i] . get Post () = = "HOD")
{
Cout <<" HOD is : "<< S[i] . accept () <<end l;
found HOD = True ;

break ;
}
}
if (!found HOD)
Cout <<" HOD not found. " <<end l;

return 0;
}
```

Output . . . .

Name: Amish
Post: Staff
Name: Aryan
Post: Head

Name: Yash
Post: Administration

HOD is:- Aryan.

# Experiment - 3

3.

a)

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

Class Book{
private:
 String title;
 String author;
 float price;

public:
Void accept (){
Cout << "Enter book title: ";
getline (cin, title);
Cout << "Enter Author name:";
getline (cin, author);
Cout << "Enter Price:";
Cin >> price;
}
Void display (){
Cout << "In Book details " << endl;
Cout << "Title:" << title << end l;
Cout << "Author:" << author << endl;
Cout << "Price:" << Price << end l;
}
};
int (){
Book b1;
Book * ptr = &b1;

ptr → accept ();
ptr → display ();
```

```
return 0;
}

//Output .....

Enter book Title: The Alchemist
Enter Author name: Paulo Coetho
Enter Price: 299.45

--- Book details ---
Enter book Title: The Alchemist
Enter Author name: Paulo Coetho
Enter Price: 299.45
```

b)

```
#include<iostream>
using namespace std;
class student {
private:
int roll_no;
float percentage;

public:
void accept (){
Cout <<" Enter roll number:"};
Cin >> this -> roll_no;

Cout <<" Enter Percentage:"};
Cin >> this -> percentage;
}
Void display (){
Cout <<" Roll Number: "<< this -> roll_no <<endl;
Cout <<" Percentage: " << this -> percentage << "%"<<endl
```

```
};
int main (){
Student s;
s.accept ()
s.display()

return 0;
}

//output ....

Enter roll Number : 101
Enter percentage : 85.8
Enter roll no: 11
Percentage : 85.8 %
```

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

Class outer {
public :
void displayouter (){

Cout <<" This is the outer class." <<endl;
}
class Inner {
public :
void displayInner (){
Cout <<" This is the Inner (nested) class." <<endl;
                                              endl;
}
};
```

```
};
int main (){

Outer outerobj;
OuterObj. displayOuter ();

Outer :: Inner innerobj;
inner obj. displayInner ();

return 0;
}

//output . . . .

This is the outer class.
This is the Inner (nested) class.
```

5/8/25

# Experiment -04

Q.1)

→

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

class Number {
  int value;

public:
  //constructor to initialize value

  Number(int v=0){
    value = v;
}
// Member function to display value

  void display (){
  cout << "value: " << value << endl;
  }
  //member function to swap values

  void swap (Number &obj){
  int temp = value;
  value = obj.value;
    obj.value = temp;
  }
};
  int main (){
  Number num1(10), num2(20);

  cout << "Before Swap:"
```

```
Cout << " num1: "; num1. display ();
Cout << "num2:"; num2. display ();

// swap values using member function

num 1. swap (num 2);

Cout << "\n After swap: " << endl;
Cout << "num1:"; num1 . display ();
Cout << "num2:"; num2 . display ();

return 0;
}
```

2.
```
#include<iostream>
using namespace std;

class number {
private:
int value;

Public:

//construction to initialize value
Number (int v){
  value =v;
}
//friend function declaration
friend void swapNumber (Number &n1,
Number &n2);

//function to display the value
   void display (){
Cout <<"value:" << value <<endl;
}
};
// friend function definition
void swapNumbers (Number &n1, Number &n2){
  int temp = n1.value;
  n1.value = n2.value;
  n2.value = temp;

int main (){

Number num1 (10);
Number num2 (20);
```

```
Cout << " Before swapping: " << endl;

Cout << " num 1 -> ";
num 1. display ();
cout << " num 2 -> " ;
num2. display ();

SwapNumbers (num 1, num2);

Cout << " In After swapping : " << endl;
 cout << " num 1 -> " ;
 num1. display ();
 Cout << " num -> " ;

 num2. display ();

 return 0;
}
```

3.
```
#include <iostream>
Using namespace std;

Class class B; // foward declaration

class class A {
private
 int numA;

Public
 class A (int a) {

 NumA = a;
 }
 void display () {
 Cout << "value in class A : " << numA << endl;
 }
 // Declare friend function
 friend void swapvalues (class A &a ; class B
  & b);
 };
 class class B {
 private:
  int numB;
 Public:
  class B (int B) {
   numB = b;
  }
  void display () {
  Cout << "value in class B : " << numB << endl;
  }
// Declare friend function
```

```
friend void swapvalues (class A &a; classB &b
);

//friend function definition

void swapvalues (class A &a , class B &b){
int temp = a.numA;
a.numA = b.numB;
b.numB = temp;
}
int main (){
class A objA(10);
class B ObjB(20);

Cout <<"Before swapping:"<<endl;

ObjA.display();
objB.display();

Swapvalues (objA, objB);

Cout <<"\n After swapping:"<<endl;
objA.display();
objB.display();

return 0;
}
```

A {a; classB {b

ass B {b){

endl;

<< endl;

4.

```
Class result1 :
def - init - (self);
Self . marks = 0

def read - marks (self):
self . marks = float (input ("Enter marks for
result 1: "))

class Result 2;
def _ init __ (self):
 self . marks = 0

def read _ marks (self):
self . marks = float (input ("Enter marks for result2:")

# Create objects of both classes

 Student 1 = result1 ()
 Student 2 = result2 ()

# calculate average
average = (Student 1 . marks + student 2 . marks) /2

# Display result
printf (f"\nAverage of the two result : {average}")
```

5.

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

class B; // forward declaration

Class ClassA {
private:
  int numA;

public:
void read data (){
Cout <<"Enter number for ClassA: ";
 Cin >> num A;
 }
friend void findGreatest (Class A, class B);
};
class ClassB {
private:
 int numB;

Public:
void readData (){
Cout <<"Enter number for Class B: ";
 Cin >> numB;
}
friend void findGreatest (Class A, class B){
 if (a.numA > b. numB) {
 Cout <<"Greatest number is: "<<a.numA <<endl;
}else if (b. numB > a. numA ) {
Cout <<"Greatest number is: "<<b.numB <<endl;
}else{
Cout << " Both numbers are equal. " <<endl;
```

```
}
}
int r
class
Class

Obj A
obj B

find

retu
}
```

```
}
}
int main(){
Class A objA;
Class B objB;

ObjA. readData();
objB. readData();

findGreatest(objA, objB);

return 0;
}
```

* Practice Question on friend function

Q.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 sum.

Code:

```
# 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 oB);
} na;
 Class class B {
    int B;
 Public;
  Void accept (){
  Cout << "Enter value for B:";
    Cin >> B;
}
friend int sum (ClassA oA, Class B oB);
} nb;
int sum (Class A 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
```

Q.2)

Code:

```
#include <iostream.h>
using namespace std;
class Number{
   int num;

Public;
 void accept (){
 Cout <<" Enter value :";
  Cin >> num;
}
void display (){
Cout <<"Enter value :";
```

```
    Cin >> num;
}
void display (){
Cout << "value" : <num << endl;
}
 friend void Swap number (number & num1, number
 & num2);
}
n1, n2;
Void swapnumbers (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 ();
    return 0;
}

//output

Enter value : 3
enter value : 6

Before swap:

    value: 3
    value: 6
```

Q.3)

After swap:

value: 6
value: 3

Q.3)

Code:

```
#include<iostream.h>
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 find integer findgreater (Box b, Cube C);
};
   Class cube{
    double V;

Public
   void accept (){
   double s;
Cout<<"Enter side length for the cube:";
```

Q.4)

```
Cin >> s;

V = s * s * s;
} friend void findGreater (box B, cube C);
} C;

void find Greater (Box, Cube) {
    String res = (b.v > c.v) ? "Box" : ((c.v > b.v) ?

                "Cube" : "Both equal");

Cout << "Greater Volume": << res << endl;
}
int main () {
    b.accept ();
    C.accept ();
    find greater (b, c);

return 0;
}
```

Output:

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

Q.4)

Code:

```
#include<iostream.h>
using namespace std;
class complex {
double r, i;

Public:
void accept (){
Cout <<" Enter real and imaginary parts : ";
Cin >> r >>i;
}
void ~~accept~~ display (){
Cout <<r<< " + " <<i <<" i " << end l;

}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;
}
```

Q.5)

# Code:

```
#include<iostream.h>
using namespace std;
Class Student {
string n;
int m [3];

Public;
void accept (){
Cout << "Enter marks for from three subjects";
Cin >> m [0] >> m [1] >> m [2];
}friendvoid calculatingAvg. (Student s);
} s;

Void calculate Average (Student) {

    double avg = (s.m [0]+s.m[1]+s.m [2]
                    3.0;

  Cout <<"Student :" s.n << endl;
  Cout << "Average Marks :" <<avg << endl;
}
int main (){
s. accept } ();
calculate average (s);

return 0;
}
```

Q.6)

Code:

```
#inclu
Using
class
class
class

int

V
Cout
Cin
} fri
Cla
int
voi
Cou
C
} fr
}e
int
re
int
a
b
c

Cou

ret
}
```

Q.6)

Code:

```
#include<stdio.h>  #include<iostream.h>
Using namespace std;
class Beta;
class Gamma;
class Alpha{

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)<<end l;

return 0;
}
```

Q.7)

Code:

```
#include<iostream>
using namespace std;
#include<cmath>
Class point {
double x,y;

Public:
void accept (){
Cout <<" Enter x and 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;
}
```

Q.8)

Code:

```
#include<stdio.h><iostream.h>
using namespace std;
class BankAccount;
class Audit {
Public;
 void pBal (Bank account);
};and;
class Bank account {
double b; public:
  void accept () {
Cout << "Enter Bank account balance" :;
 Cin >> b;
}friend void audit: pbal (BankAcc);
};ba;
Void . Audit :: pBal (Bank Account) {
Cout << << "Auditing . . . . BankAcc Balance : ₹ "
 << Ba . b << end l;
}
int main () {
ba . accept ();
and . pbal (ba);

return 0;
}
```

# Experiment - 05

Q.1) a) Write a program 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, sum;
   Public:
    Sum (int num)
    {
      n=num;
      sum=0;
      for (int i=0; i<n; i++)
         {
            sum += i;
         }
      cout << sum = << sum;
    }
};
int main ()
{
  Sum S(10);
  return 0;
}
```

Output:

Sum : 55

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 & display data for one student.

```
#include <iostream>
 include <string>
using namespace std;
class student
{
  private:
  string name;
  float percentage;
  public:
   Student (string n, float p)
   {
     name = n;
     percentage = p;
     cout << "Name : ";
     getline (cin, name);
     cout << " Percentage : ";
      cin >> percentage;
   }
   void Display ()
   {
   cout << "Student Name: " << name << endl;
   Cout << "Student Percentage: " << percentage << endl;
   }
};
```

```
int main()
{
  Student S1("Tanay", 92.8);
  S1.display();
  return 0;
}
```

Output:

Name: Tanay
Percentage: 92.8
Student Name: Tanay
Student Percentage: 92.8

C) Define a class college members: variables 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 <string>
#include <iostream>
using namespace std;
class College
{
  private:
    String course_name, stud_name;
    int roll_no;
  Public:
    College()
    {
      roll_no = 19;
      stud_name = "Tanay";
      course_name = "AI&DS";
    }
```

```
void display ()
{
cout<<"Roll no: " << roll_no << endl;
cout << "student_name: "<< stud_name <<endl;
cout <<" course name: " <<course_name <<endl;
}

int main ()
{
  college c1;
  c1. display ();
  return 0;
}
```

Output:
Roll no : 19
Student name: Tanay
Course name: AI & DS

d) Write a program to demonstrate constructor overloading.

```
# include <iostream>
using namespace std;
class rectangle
{
   int l, b;
Public:
  rectangle ()
{
   l = 3;
   b = 6;
}
```

```
rectangle (int x)
{
   l = x;
   b = x;
}

rectangle (int x, int y)
{
   l = x;
   b = y;
}
rectangle (rectangle & r3)
{
   l = r3.l;
   b = r3.b;
}

Void calculate ()
{
cout << "The area is : " << (l * b) << endl;
}
};

int main ()
{
   rectangle r1;
   r1.calculate ();
   rectangle r2 (3);
~~r2rectangle~~
   r2.calculate ();
   rectangle .r3 (6, 6);
   r3.calculate ();
```

```
    rectangle r4 (r3);
    r4.calculate ();
    return 0;
}
```

Output:

```
The area is : 18
The area is : 9
The area is : 36
The area is : 36
```

14/10/25

# Experiment -6

a) Write a program to implement multi level inheritance.
Assume suitable data

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

 Class Student: Public Person
 {
  Private:
   int roll_no;
  Public:
 void accept ()
 {
   cout << "Enter your name: ";
   cin >> name;
   Cout << "Enter your age: ";
   cin >> age;
   Cout <<" Enter your roll no: ";
   Cin >> roll_no;

 }
```

```
void display ()
{
  cout <<" Name : \n " << name;
  Cout <<" Age : \n " <<age;
  Cout <<" Roll no : \n " <<roll_no;
 }
};

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

Output :

Enter your name : Tanay
Enter your age : 19
Enter your roll number : 49

Name: Tanay
Age: 19
Roll No.: 49

b) Write a program to implement multiple

```
#include<iostream>
using namespace std;
class Academic
{
  Protected:
  int marks;
};

Class Sports
{
  Protected:
  int score;
};

Class Result: Protected Academic, Protected Sports
{
 int total_score =0;
 Public:
 void accept()
 {
 Cout << "Enter the marks of the student: "
 Cin >> marks;
 Cout << "Enter the sports score of the
          student: ";
  Cin >> score;
 }
```

```
void calculate ()
{
total_score = marks + score;
Cout <<"The marks of the student is: "<<marks
                                    <<endl;
Cout <<" The sports score of the student is: "<<score
                                    <<endl;
Cout <<" The total score of the student is: "
                                    <<total_score <<endl;
}
};

int main ()
{
  Result r;
  r.accept ();
  r.calculate ();
  return 0;
}
```

Output:

```
Enter the marks of the student : 86
Enter the sports score of the student: 90

The marks of the student is : 86
The sports score of the student is : 90
The total score of the student : 176
```

c) Write a program to implement hirarchical inhevetance. Assume suitable data

```
#include <iostream>
using namespace std;
Class vehical
{
  protected:
  String brand;
  int model;
};

Class car: Protected Vehical
{ Protected:
   String type;
};

Class Electric car: protected car
{
 int batterycapacity;
 Public:
 Void accept ()
{
 cout << "Enter the brand of car: ";
 cin >> brand;
 cout << "Enter the model of car: ";
 cin >> model;
 Cout << "Enter the type of car: ";
 cin >> type;
 Cout << "Enter the battery capacity of car: ";
 Cin >> battery capacity;
}
```

```
void display ()
{
Cout <<"\n The brand of the car: " <<brand<<endl;
Cout<<" The model of car: "<<model<<endl;
Cout<<" The type of car: "<<type<<endl;
Cout<<" The battery capacity of car: "<<battery
                                    capacity <<endl;
}
};
int main ()
{
  Electric car e;
  e.accept ();
  e.display ();
  return 0;
}
```

Output:

```
Enter the brand of car: Hyundai
Enter the model of car: Venue
Enter the type of car: SUV
Enter the battery capacity of car: 52.4 KWH
```

d) Write a program to implement hybrid inheritance

```
# include <string>
# include <iostream>
 using namespace std;
Class Person
{
  Public:
  String name;
  int age;

 Void getPersonDetails ()
 {
   Cout << " Enter name: ";
   Cin >> name;
   Cout << "Enter age: ";
   Cin >> age;
 }
 Void showDetails ()
{
Cout <<"Name " <<name << "Age: " <<age <<endl;
}
};
 Class Student: Public Person
 {
   Public:
   String course:
   Void get Details ()
{
   Cout << "Enter course: ";
   Cin >> course;
}
```

```
  void showDetails()
  {
    cout << "course: " << course << endl;
  }
};

Class Employee : Public Person
{
  Public:
  ~~void get~~
    String Company;
    void getEmployeeDetails()
    {
      cout << "Enter company: ";
      cin >> company;
    }

    Void showEmployeeDetails()
    {
    cout << "Company: " << company << endl;
    }
};
Class Intern: Public student, Public Employee
{
  Public:
      Void showInternDetails()
    {
     cout << "\n--- Intern Details ---\n";
     student: showPersonDetails();
     showEmployee Details();
    }
};
```

```
int main()
{
 Intern it;
 it.Student::getPersonDetails();
 it.getDetails();
 it.getEmployeeDetails();
 it.showInternDetails();

 return 0;
}
```

Output:

```
Enter name: Tanay
Enter Age: 17
Enter Course: AI & DS
Enter Company: TCS

---Intern Details---

Name: Tanay
Age: 17
Course: AI & DS
Company: TCS
```

14/10/25

# Experiment - 7

a) Write a program using function overloading to calculate the area of a claboratory (which is rectangular in shape) & area of classroom.

```
#include<iostream>
using namespace std;
class Area;
{
  Private:
  Float l, b;
  Public:
  void area (float l)
  {
    Float area;
    area = l * b
    cout << "Area of Square: " << area << endl;
  }
  void area (float l, float b)
  {
    float area;
    area = l * b;
    cout << "Area of Rectangle: " << area;
  }
};
int main()
{
  Area a1;
  a1. area (8);
  a1. area (9, 12);
  return 0;
}
```

Output:

Area of Square : 64
Area of rectangle : 48

b) Write a program using function overloading to calculate the sum of 5 float values & 10 int.

```
# include <iostream>
  using namespace std;
  class sum
{
  Private:
  int a, b, c, d, e, f, g, h, i, j;
  float k, l, m, n, o;

  Public:
  void sum (float k, float l, float m, float n, float o)
{
   float sum;
   sum = k+l+m+n+o;
   cout <<"Sum of 5 floating numbers: "<<sum
                                  <<endl;
  }
  void sum(int a, int b, int c, int d, int e, int f,
        int g, int h, int i, int j)
{
   int sum;
   sum = a+b+c+d+e+f+g+h+i+j;
   cout <<"sum of 10 integers: "<<sum;
}
};
```

```
int main ()
{
 Sum S1;
 S1. sum (1.2, 3.5, 5.6, 8.4, 1.5);
 S1. Sum (1,2,3,4,5,6,7,8,9,10);
 return 0;
 }
```

Output:
Sum of 5 floating numbers: 20.2
Sum of 10 integers : 55

c) Write a program to implement unary operator when used with the obj so that the numeric date member of the class is negated

```
#include <iostream>
 using namespace std;
 class Number
{
 Private : int x;
 Public : void accept ()
 {
  cout << " Enter a number : ";
  Cin >> x;

 Void operator- ()
{
   x = -x
}
```

```
void display ()
{
  cout <<"Negated number: "<<x<<endl;
 }
};

int main ()
{
  Number n1;
  n1.accept ();
  -n1;
  n1.display ();
  return 0;
 }
```

Output:

```
Enter a number: 5
Negated number: -5
```

d.) Write a program to implement the unary ++ operator (for pre increment and post increment) when used with the object so that the numeric data member of the class is incremented.

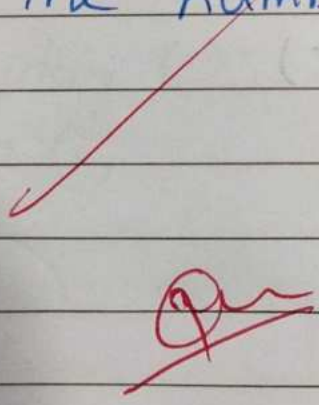
```
#include <iostream>
 using namespace std;
 class Number
 {
  private: int x; int accept;
  public: void accept ()
  {
```

```
cout <<"Enter a number: "; cin >> x;
temp = x;
      }
void operator ++ ()
 {
    x = ++x;
 }
void reset ()
{
   x = temp;
 }
void operator ++ (int)
{
  x = x++;
}
void display ()
{
 cout <<"(pre) The number is: " << x << endl;
}

Void display 2 ()
{
 cout <<"(post) The number is: " << x;
 }
};

int main ()
{
 Number n1;
 n1. accept ();
```

```
    n1.display();
    n1.reset();
    n1++;
    n1.display2();
    return 0;
}
```

Output:

```
Enter a number : 5
(pre) The number is: 6
(post) The number is: 5
```



# Experiment - 8

a) Write a program to overload '+' operator so that two strings can be concantenated. Eg.: XYZ+PQR = XYZPQR

```
#include <iostream>
#include <string>
using namespace std;
class mystring
{
  public: string str;
  mystring operator +(mystring other)
  {
    mystring temp;
    tempstr = str + other.str;
    return temp;
  }
  void display()
  {
    cout << str << endl;
  }
};
int main()
{
  mystring s1,s2,s3;
  s1.str = "XYZ"
  s2.str = "pqr"
  s3 = s1 + s2
  cout << "Concatenated string: ";
  s3.display();
  return 0;
}
```

Output:

Concantenated string: XYZPQR

b)

```
#include <string>
#include <iostream>
using namespace std;
class login
{
  Protected: string name, password;
  Public: virtual void accept ()
{
   cout << "Enter name: ";
   cin >> name;
   cout << "password: ";
   cin >> password: ;
}
};
  class Emaillogin : Public login {
          string email;
      Public:
      void accept () override {
       Login : accept ()
       cout << "Enter email: ";
       cin >> email;
      }
    void display ()
{
   cout << "\n --- Email login Details --- \n".
   cout << "Name: " << name << "password: " <<
   password << "Email: " << email << endl;
}
};
```

```
class Membershiplogin : Public login
    {
     sting membership ID;
    Public:
     Void accept ()
     override
  {
    Login : accept ();
    Cout << " Enter membership ID : ";
    Cin >> membershipID;
  }

  Void display ()
  {
  Cout <<"\n --- Membership Login Details ---\n";
  Cout <<" Name : "<<name <<"\npassword : "<< password<<
   "Membership ID : "<< membership ID <<endl;
  }
};
  int main ()
  {
    Emaillogin e;
    Membership login m;
    e. accept ();
    m. accept ();
    e. display ();
    m. display ();
    return 0;
  }
```

Output

```
Enter Name: Tanay
Enter Password: 4567
Enter Email: tanay@gmail.com
Enter Name : Tanay
Password : 8901
Enter membership ID: 1256
--- Email Login Details ---
    Name : Tanay
    Password : 4567
    Email: tanay7@gmail.com
--- Membership Login Details ---
    Name : Tanay
    password : 8901
    MembershipID : 1256
```

Experiment NO:-9

* Write a program to perform various operations on file.

1) Write a program to copy the contents of one file into another. Open "First" in read (ios::in) mode & 'second.txt" file in write (ios::out) mode. Copy contents of "First.txt" in "Second.txt" Assume "1st.txt" is already created.

```
#include <iostream>
#include <fstream> // file handling
using namespace std;

int main () {
    // Input filestream (read mode)
    ifstream inFile ("First.txt", ios::in);

    // Check if input file opened successfully
    if (!inFile) {
        cout << " error : cannot open First.txt
                for reading! << endl;
        return 1;
    }

    // output filestream (write mode)
    ofstream outFile ("Second.txt", ios::out);

    // Check if output file opened successfully
    if (!outFile) {
        cout << "Error: Cannot open Second.
                txt for writing!" << endl;
```

```
    inFile.close();
    return 1;
}
    char ch;
    // Read character by character from
    First.txt & write to second text.
    while (inFile.get(ch)) {
        outFile.put(ch);
    }
    cout << " File copied successfully!" << endl;

    // Close both files
    inFile.close();
    outFile.close();

    return 0;
}
```

2) Write a C++ program to count digits & spaces using file handling.

```
#include <iostream>
#include <fstream>
#include <cctype>

using namespace std;

int main() {
    string filename;
    cout << " Enter the filename: ";
    cin >> filename;
```

```
ifstream file(filename); // open file for reading
if (file) {
    cerr << " Error: Could not open file" << endl;
    return 1;
}
char ch;
int digit count = 0;
int space count = 0;

// Read the file character by character.
while (file.get(ch)) {
    if (isdigit(static_cast<unsigned char>(ch))) // digit count
    else if (isspace(static_cast<unsigned char>(ch)))
        space count++;
}

file.close();

cout << " \n Number of digits: " << digit count << endl;
cout << " \n Number of spaces: " << space count << endl;

return 0;
}
```

Q3) Write a C++ count words using File Handling.

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {
    string filename;
    cout << "Enter the filename: ";
    cin >> filename;

    ifstream file(filename); open
                        file for reading.

    if (!file) {
        cerr << "Error: Could not open
                file" << filename <<
                "!!!. !!!!
        return 1;
    }
    string word;
    int word count = 0;

    // Read each word from the file
    while (file >> word) {
        word count++;
    }

    file.close(); // Close the file

    cout << "Total no of words in the
            file: " << word count << endl;
    return 0;
}
```

(Q34)

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

int main() {
    ifstream file;
    char filename[50];
    char ch;
    int digitcount=0, spacecount=0;

    cout << "enter the file name:";
    cin >> filename;

    file.open(filename);

    if(!file) {
        cout << "Error opening file!" << endl;
        return 1;
    }

    while(file.get(ch)) {
        if(ch>='0' && ch<='9')
            digitcount++;
        if(ch==' ')
            spacecount++;
    }

    file.close();
    cout << "Total Digits: " << digitcount
         << endl;
    cout << "Total Spaces " << space << endl;

    return 0;
}
```

Q5]

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

int main() {
    ifstream fin("First.txt");
    if (!fin) {
        cout << "File not found!";
        return 0;
    }

    string search, word;
    int count=0;

    cout << "Enter word to search";
    cin >> search;

    cout << "occurence of '" << search << "' is " <<
        count << endl;
    fin.close();
    return 0;
}
```

Exp 10 -

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

template <class T>
T sumArray(T arr[], int n){
    T sum=0;
    for(int i=0; i<n; i++)
        sum += arr[i];
    return 0;
}

int main(){
    int a[5] = {1,2,3,4,5};
    float b[5] = {1.1, 2.2, 3.3, 4.4, 5.5};
    double c[5] = {1.11, 2.22, 3.33,
                   4.44, 5.55};

    cout<<"Sum of int arrays: "<<sum
        Array(a,5)<<endl;

    return 0;
}
```

b)
```
#include <iostream>
#include <string>
using namespace std;

template <class T>
T square(T x){
    return x*x;
}
```

```
template()
String square (string s) {
    return s+s; // String concatenation with itself.
}
int main() {
    int a = 5;
    string str = "ABC";
    cout << "Square of int: " <<
        square(a) << endl;
    return 0;
}
```

c)

```
#include <iostream>
using namespace std;
template <class T>
T a, b;
public:
calculate (T x, T y) {
    a = x;
    b = y;
}
void add() cout << a+b << endl;
void sub() cout << a-b << endl;
int main() {
calculate <int> obj1(10,5);
calculate <float> obj2(5.5,2.2);
```

```
cout << "\n-- Float Calculator --"
     << endl;
obj2.add();
obj2.sub();

obj2.mul();
obj2.div();
return 0;
}
```

d)

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

template <class T>
class Stack {
    T s[20];
    int top;
public:
    Stack() { top = -1; }

    void push(T x) {
        if (top == 5)
            cout << "Stack Overflow!" << endl;
        else
            s[++top] = x;
    }
    void pop() {
        if (top == -1)
            cout << "Stack Underflow!"
        else
            cout << "Popped: " << s[top--]
            endl;
    return 0;
}
```

Exp 11

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

template <typename T>
class GenericVector {
private:
    std::vector<T> data;
public:
    // create using size (default is initialized
    GenericVector(size_t n = 0) : data(n) {}

    // create from initializer list
    GenericVector(initializer_list<T> il) : data(il) {}

int main() {
    // create int vector
    GenericVector<int> gv{10, 20, 30, 40};

    gv.display();
    cout << "\n";

    // modify element
    cout << "After modification:\n";
    gv[2] = 300;
```

// multiply by scalar.
gv.multiply_by_scalar(2);
cout << "\n\n";

// create double vector
Generic vector <double> gv2{1.5, 2.5, 3.5};
gv.display();

cout << "After multiply by 2.0";
gv.2 display();
cout << "\n";

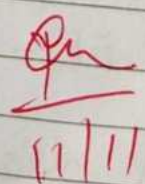
cerr << "Error:" << e.what() << "\n";

return 0;
}

Experiment - 12

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

int main() {
stack <int> st;

st.push(10);
st.push(20);
st.push(30);

cout << "stack contents (top to bottom):
stack <int> temp = st;
while (!temp.empty()) {
cout << temp.top() << " ";
temp.pop();
}
cout << "\n";

cout << "Top element: " << st.top()
<< "\n";
st.pop();

cout << "After pop, new top: " << st.empty()
? -1 : st.top() << "\n";
```

```
cout << "stack size: " << st.size() << "\n";
cout << "Is Stack empty! " << st.empty() ?
    "yes" : "No") << "\n"
return 0;
}
```



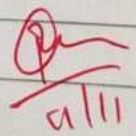
Ex: 11 - (b)
with Iterators

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

template <class T>

class Myvector {

  vector <T> v = {10, 20, 30};

  Public :
    void modify (int, i T val)
{
  v[i] = val;
}
void multiply (T s) {
for (typename vector <T> :: iterators it =
      v.begin();

   it ! = v.end();
   ++ it)
    * it = (*it) * s;
}
Void display () {
  Cout << "( ";
for (typename vector <T> :: iterator it
  = v.begin.() ; it ! = v.end();
     ++ it) {
```

```
        cout << *it;
        if (it != v.end() - 1);
        cout << ", ";
        }
        cout << ")\n";
    }
};
int main(){
    MyVector <int> v;
    v.display();
    v.modify(1,50);
    v.display();
    v.multiply(2);
    v.display();
}
```

Output:

(10,20,30)
(10,50,30)
(20,100,60)



# Expt - 12

b)
```
#include <iostream>
#include <Queue>

Using namespace std;

int main (){

    Queue<int> q;
    q. push (10);
    q. push (20);
    q. push (30);

cout << "Queue elements <front to rear): ";
while (!q. empty ()){
        cout <<q. front () << " ";
        q. pop ();
    }
}.
```

c)
```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct person {
    string name;
    string birthdate;
    string phone;
};

bool compareByName(const person& a,
                   const person& b) {
    return a.name < b.name;
}

int main() {
    vector<person> people = {{"Alice", "01-01-2000",
                              "111"},
    {"bob", "02-02-1999", "222"}, {"Charlie",
     "03-03-2000", "333"}};
    sort(people.begin(), people.end(), compareByName);

    cout << p.name << " " << p.birthDate
         << " " << p.phone << "\n";
```

```
String searchName = "Bob";

auto it = find_if (people.begin(), people.end()
          [&](person &p){ return p.name ==
            searchName;
};
if(it != people.end())

cout << "Found: " << it->name << " it->
  birthdate << " " << it->
  phone << "\n";

else
  cout << "\nPerson not found\n";
}
```