Experiment No:-1

1) WAP to declare a class student having data members as name, Roll-no. Accept & display data for one student.

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

class student {
        string name;
        int roll no;

public:
    void acceptance () {
        cout << "Enter student's name:";
        getline (cin, name);
        cout << "Enter roll number:";
        cin >> roll no;

    }
    void display data () {
        cout << "| Enter student's name <<endl;
        cout << "Name:" << name << endl.
        cout << Roll NO:" << roll NO << end l;

} ;}

int main () {
    S1. accept data ();
    S1. display data ();
    return 0;
}
```

Output:-

Enter student's name:- Tejas Katkar
enter roll number:- 65

Student's details:-
Name:- Tejas Katkar
Roll No:- 65

) WAP to declare a class Book. having data members as id, name, price. Accept data for 2 books. & display data of book having greater price.

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

class Book {
    int id;
    string name;
    float price;

public:
    void accept() {
        cout << " enter book ID:";
        cin >> id;
        cout << "enter book name:";
        cin >> name;
        cout << "enter book price:";
        cin >> price;
    }
    void display() {
        cout << "Book ID" << id << endl;
        cout << " Book name" << name << endl;
        cout << " Book price:" << price << endl;
    }
    void display() {
        cout << " Book ID" << id << endl;
        cout << " Book name" << name << endl;
        cout << " Book price:" Price << endl;
    }
    float get price() {
        return price;
    }
};
int main() {
    int n;
    cout << "enter number of book:";
    cin >> n;

    Book books[n];

    for (int i=0; i<n; j++) {
    cout << "\n enter details of books"
        << j << " :" << end end);
    books[i] . accept();
}
    int max Index=0;
    for (int i=1; i<n; i++) {
    if (books[i] . get Price() >> books
        [max Index] . get Price())

} cout << "\n Book with greatest
        Price:" << end1;
```

books [max Index].display ();
return 0;
}

Output:-

Enter number of books :- 2

Enter details of book 1
Enter book ID :- 6565
Enter book Name :- kii
Enter book price :- 987

Enter details of book :- 2
_____ " _____ :- 45
_____ " _____ :- dsy
_____ " _____ :- TSK
_____ " _____ :- 65

WAP to declare a class time having data members as H, M & S. Accept data for one object & display total time in sewnas.

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

class Time {
private:
    int H, M, S;

public:
// Default constructor
Time () : H (0), M (0), S (0) {}

// Method to read time from user
void display Total seconds () const {
    cout << "Total time in seconds = "
         << to seconds () << " seconds";
}
};
int main () {
    Time t;
    t.read ();
    t.display Total seconds ();
    return 0;
}
```

Output

Enter hours :- 23
Enter minutes :- 43
Enter seconds :- 67
Total time in seconds :- 85447 seconds

23/7/23

Exp:- 2

* Write a C++ program to demonstrate the use of array of objects.

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

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

class city {
public:
  string name;
  int population;

  void get data () {
    cout << "Enter city name: ";
    cin >> ws; //
    getline (cin, name);
    cout << ("Enter population of " <<name;
    cin population
  }
};

int main () {
  const int num cities = 5;
  city cities [num Cities];
```

```
// Accept data for 5 cities
for (int i=0; i< num Cities; i++) {
  cout << "\n Enter details for city" << i+1 <<
       " :\n";
  cities [i] . get data ();
}
// Find city with highest population
int max Index = 0;
for (int i = 1; i < num cities; i++) {
  if (cities [i]. population > cities
      [ max Index]. population) {
    max Index = i;
  }
}
// Display city with highest population
(int max Index = 0;
for (int i = 1; i < num cities ; i++) {
  if (cities [i]. population > cities [max Index].
      population) {})
cout << "\n city with highest population
     " cities [max Index] . name;
cout << " with a population of " << citi
     [max Index]. population << ". " <<

return 0;
}
```

Output:-

```
Enter details for city 1:
Enter city name: Pune
Enter population of Pune: 34
```

Enter details for city 2:-
Enter city name:- Delhi
Enter population of Delhi:- 24

Enter details for city 3:-
Enter city name:- Nashik
Enter population of Nashik:- 56

Enter details for city 4:-
Enter city name:- Nagpur
Enter population of Nagpur:- 87

Enter details for city 5:-
Enter city name:- Jalgaon
Enter population of Jalgaon:- 78.

Output:-

The city with highest population is Jalgaon.

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

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

class account{
private:
    int accNo;
    float balance;
public:
    // Function to accept account data
    void input(){
        cout << "Enter Account Number:";
        cin >> accNo;
        cout << "Enter Balance:";
        cin >> balance;
    }
    // Function to apply 10% interest if balance>=5
    void applyInterest(){
        if(balance >= 5000){
            balance += balance * 0.10; // Add 10% inte
        }
    }
    // Function display account info
    void display(){
        if(balance >= 5000){
            cout << "Account No" << accNo << ",
            Balance after interest:" << balance <<
        }
    }
};
int main(){
    Account accounts[10];

    // Accept data for 10 accounts
    cout << "Enter details for 10 accounts:\
    for(int i=0; i<10; i++){
        cout << "\nAccount" << i+1 << ":\n
        accounts[i].input();

    // Apply interest
```

```
for (int i=0; i<10; i++) {
    accounts[i].applyInterest();
}
// Display accounts with balance >= 5000 (after interest)
cout << "\n Accounts with balance >= 5000 after adding interest:\n";
for (int i=0; i<10; i++) {
    accounts[i].display();
}
return 0;
}
```

Output:-

Enter details for 10 accounts

Account 1:-
Enter Account Number:-3456
Enter Balance:-45

Account 2:-
Enter Account Number-8899
Enter Balance:-67

Account 3:-
Enter Account Number-12345
Enter Balance:-67

Account 4:-
Enter Account Number-6790
Enter Balance:-12

Account 5:-
Enter Account Number:-4390
Enter Balance:-678

Account 6:-
Enter Account Number:-5643
Enter Balance:-9087

Account 7:-
Enter Account Number:-4532
Enter Balance:-8765

Account 8:-
Enter Account Number:-6743
Enter Balance:-900

Account 9:-
Enter Account Number:-6000
Enter Balance:-678

Account 10:-
Enter Account Number:-3742
Enter Balance:-0987

Accounts with balance >=5000 after adding interest:-

Account No 5643 Balance after interest: 9995.7
Account No 4532, Balance after interest:: 9641.

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 {
private:
  string name;
  string post;

public:
  // Function to accept staff data
  void input () {
      cout << "Enter Name:";
      getline (cin, name);
      cout << "Enter Post:";
      getline (cin, post);
  }
  // Function to check and display if post is HOD
  void display IF HOD () {
      if(post == "HOD" || post == "hod" || post == "Hod") {
          cout << "HOD Name:" << name << endl;
      }
  }
};

int main () {
    staff staffList [5];
    cout << "Enter details for 5 staff members:\n";
    for (int i = 0; i < 5; i++) {
        cout << "\n Staff " << i+1 << ":\n";
        cin.ignore (); // Clear input buffer before
          getline
        staffList [i].input ();
    }
    cout << "\n List of staff members who are
        "HOD" :\n";
    for (int i = 0; i < 5; i++) {
        staffList [i].display IF HOD ();
    }
    return 0;
}
```

output:-

```
Staff 1:
Enter name: Dr Sharma
Enter Post : HOD

Staff 2:
Enter name:- Dr Joshi
Enter Post:- Assistant Professor

Staff 3:
Enter name:- Ms Neha
Enter Post :- HOD

Staff 4:- Mr Rakesh
Enter Post:- Lecturer

Staff 5:-
Enter Name:- Prof
Enter Post:- HOD
```

28/7/25

List of Staff members who are 'HOD'

HOD Name :- Dr Sharma
HOD Name :- Ms Neha
HOD Name :- Prof. Singh

~~(Experiment No: 3)~~
~~(Experiment no: 3)~~
Practice Questions

Q1] Write a C++ program for addition of 2 numbers.

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

int main() {
    int num1, 2, sum;

    // Input two numbers
    cout << " Enter first number:";
    cin >> num1;

    cout << " enter second number:";
    cin >> num2;

    // calculate sum
    sum = num1 + num2

    // Display the result
    cout << "The sum of " << num1 << "and"
      num2 << " is " << sum << and end

    return 0;
}
```

Output:-
Enter 1st no:- 23
Enter second number:- 34
The sum of 23 & 34 is 57

2) Write a c++ program to perform arithmetic operations using switch case.

→

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

int main(){
    float num1, num2, result;
    char op;

    // Input two numbers and an operator
    cout << " Enter first number:" ;
    cin >> num1;

    cout << " Enter second number:";
    cin >> num2;

    cout << "Enter operator (+, -, *, /):";
    cin >> op;

    // Perform operation based on operator
    switch (op) {
        case '+':
            result = num1 + num2;
            cout << " Result: " << result << endl;
            break;
    Case '-':
            result = num1 - num2;
            cout << " Result: " << result << endl;
            break;
    Case '*'
            result = num1 * num2;
            cout << " Result: " << result << endl;
            break;
        case '/':
            if (num2 != 0) {
                result = num1 / num2;
                cout << " Result: " << result << endl;
            } else {
                cout << " Error: Division by zero is not
                        allowed." << endl;
            }
            break;
        default:
            cout << "Invalid operator!" << endl;
    }
    return 0;
}
```

Output:-

```
Enter 1st number:- 56
Enter second number:- 67
Enter operator (+, -, *, /): +
Result: -123
```

3) Write a c++ program to check whether a numbe even or odd.

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

Int main (){
```

```
int num;
cout << "enter an integer:";
cin >> num;

if (num % 2 == 0){
    cout << num << " is even." << endl;
} else {
    cout << num << " is odd." << endl;
}
return 0;
}
```

output:-

Enter an integer:- 45
45 is odd.

4) Write a c++ code to print 1 to 10 numbers using for loop.

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

int main(){
    for (int i=1; i<=1; i<=10; i++) {
        cout << i << endl;
    }
    return 0;
}
```

Output:-
1
2
3
4
5
6
7
8
9
10

5] Write a c++ code to print the following pattern.

1)
```
  *
 * *
* * *
```

```
#include <iostream.h>
using namespace std;
main()
{
  int rows = 3
  for (int i=1; i<=rows; i++)
  {
    for (int j=1; j<=i; j++)
    cout << "*";
    }
    cout << endl;
    }
    return 0;
}
```

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

⇒
```
#include <iostream.h>
using namespacestd;
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;
  }
```

3)
```
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 No:-3.

* Pointer to object, The 'this' pointer, nested class

1) Write a program to declare a class 'book' containing data members as book-title, author name and price. Accept & display the information for one object using a pointer to that object.

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

Class book{
private:
   string book_title;
   string author_name;
   float price;

public:
   void getDetails (){
      cout<< " Enter book title:";
      getline (cin, book_title);
      cout << "Enter author name:";
      getline (cin, author_name);
      cout << "Enter price :";
      cin>> price;
      cin.ignore ();
}

   Void display Details (){
      Cout << "| Book Details: In";
      Cout << "Title:" << book title << end l;
      Cout << " Author:" << author_name << end l;
      cout << "Price : $" << price << end l;
   }
};

int main (){
   book *ptr = new book ();

   ptr -> get details ();
   ptr -> display Details ();

   delete ptr;
   return 0;
}
```

Output:-

```
Enter book title:- English
Enter author name:- Chetan Bhagat
Enter price:- 456

Book Details:-

Title:- English
Author:- Chetan Bhagat
Price:- $456
```

2) WAP to declare a class 'student' having data mem as roll_no & percentage. Using 'this' pointer invol member functions to accept & display this data fo one object of the class.

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

class student {
private:
   int roll_no;
   float percentage;

public:
   void accept (int roll_no
   this -> roll_no = roll_no;
   this -> percentage = percentage;
  }
   void display () {
      cout << " Roll No:" << this -> roll_no <<
   }
};

int main(){
   STUDENT student1;
   // Accept data using member function
   student1. accept(65, 88.5);
   // Display data using member function
   Student1. display()

   return 0;
}
```

Output:-
Roll No:-65
Percentage:- 88.5 %.

3) Write a program to demonstrate the use of nested class.

```
#include <iostream>
using namespace std;
class student
{
public:
string name;
int roll;
void accept()
{
cout << " enter the name of student:";
cin >> name;
cout << " enter the roll number in";
cin >> roll no;
}
void display()
{
cout << " The name of student "<< name
cout << " The roll no in" << roll;
}
class marks
{
public:
int m1, m2;
void accept1()
{
cout << " enter marks of two subject
cin >> m1 >> m2;
}
void display 1()
{
float per((m1+m2) /100.0)* 100;
cout << " The percentage of 2 subject = "
<< per << endl
```

```
};
};
int main()
{
  Student S1;
  Student : marks m;
  S1.accept();
  m.accept1();
  S1.display();
  m.display1();
  return 0;
}
```

Output:-

Enter the name :- Tejas
Enter roll no: 65
Enter the name:- Tejas
The roll No:- 65

Enter 2 marks:
32
16
Percentage:- 48.

18

Experiment No:-4

1) WAP to swap 2 numbers from same class using object as function argument. Write swap function as member function.

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

class Number {
    int value;

public:
  // Function to set value
  void setvalue(int v) {
      value = v;
};

  // Function to display the value
  void display() {
      cout << "Value:" << value << endl;

  void swap(Number & obj) {
      int temp = value;
      value = obj.value;
      obj.value = temp;
  }
};

int main() {
  Number num1, num2;
```

```
// Set initial values
num1.set value (10);
num2.set value (20);

cout << "Before swap:" << endl;
cout << "num1:"; num1.display ();
cout << "num2:"; num2.display ();

num1.swap (num2);

cout << "\n After swap:" << endl;
cout << "num1:"; num1.display ();
cout << "num 2:"; num2.display ();

return 0;
}
```

Output:-

```
Before swap:
num1.value: 10
num2.value: 20

After swap:
num1.value: 20
num2.value: 10
```

WAP to swap to swap two numbers from the same class using the concept of a friend function.

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

class number {
    int value;

public:
    // Function to set value
    void set value (int v) {
        value = v;
    }

    // Function to display value
    void display () {
        cout << "Value:" << value << endl;
    }
    // Declare friend function
    friend void swap (Number &a, Number &
};

// Friend function definition
void swap (Number &a, Number &b);
    int temp = a.value;
    a.value = b.value;
    b.value = temp;
}

int main () {
    Number num1, num2;

    // Set initial values
    num1.set values (5);
    num2.set values (15);
```

```
cout << "Before swap:" << endl;
cout << "num1:num2"; num1.display();
cout << "num2:"; num2.display();

// call the friend swap function
swap (num1, num2);

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

return 0;
}
```

Output:-

Before swap:
num1 : value: 5
num2 : value: 15

After swap:
num1: value: 15
num2: value: 5

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

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

// Forward declaration
class classB;

class ClassA {
    int valueA;

public:
    void setValue(int v) {
        valueA = v;
    }

    void display() {
        cout << "Value in ClassA:" << valueA << endl;
    }
    // Declare friend function
    friend void swap(classA &, classB &);
};

class ClassB {
    int valueB;

public:
    void setValue(int v) {
        valueB = v;
    }
    void display() {
        cout << "Value in classB:" << valueB << endl;
    }
    // Declare friend function
    friend void swap(classA &, classB &);
};

// Friend function to swap values
void swap(classA &a, classB &b) {
```

```
    int temp = a.valueA;
    a.valueA = b.valueB;
    b.valueB = tempB;
}
int main(){
    classA objA;
    classB objB;

    objA.setvalue(10);
    objB.setvalue(20);

cout<<"Before swap:"<<endl;
objA.display();
objB.display();

// Call the friend function to swap
swap(objA, objB);

cout<<"\nAfter swap:"<<endl;
objA.display();
objB.display();

    return 0;
}
```

Output:-

```
Before swap:
Value in class A : 10
Value in classB : 20
After swap:
Value in class A : 20
Value in class B : 10
```

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

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

// Forward declaration
class Result2;

class Result1{
    float marks1;

public:
    void readmarks(){
        cout<<"enter marks for Result1:";
        cin>>marks1;
    }
    void display(){
        cout<<"marks in Result1:"<<marks1<<endl;
    }
    // Declare friend function
    friend float computeaverage(Result1 &r1, Result2 &r2);
};

class Result2{
    float marks2;
```

```
Public:
    Void readmarks(){
        cout<< " Enter marks for Result 2.:";
        cin >> marks2;
    }
    Void display (){
        cout << " marks in Result 2: "<< marks2<<endl;
    }
    // Declare friend function
    friend float compute average (Result1, Result2);
};

// Friend function to compute average
float compute average (Result1 r1, Result r2){
    return (r1.Marks1, + r2.marks2)/2.0;
}

int main (){
    Result1 obj1;
    Result2 obj2;

    obj1.read marks ();
    obj2. read marks ();

    obj1.display ();
    obj2. display ();

    float avg = compute Average (obj1, obj 2);
    Cout << "Average of marks:" << avg << endl;

    return 0;
}
```

Output:-

```
Enter marks for Result1:- 85
Enter marks for Result2:-75
marks in Result1:- 85
marks in Result2:- 75

Average of marks:- 80.
```

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

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

// Forward declaration
class class B;

class class A {
      int num A;

Public:
   void set Data (int a) {
        num A = a;
   }
   // Declare friend function
   friend void find Greatest (class A, class B);

};
```

```
class classB {
    int numB;

public:
    void setdata (int b) {
        numB = b;
    }
    // Declare friend function
    friend void findGreatest (class A, class B);
};

void findGreatest (class A a, class B 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 << e
    } else {
        cout << "Both numbers are equal: " << a.numA
                                        << endl;
    }
}
int main () {
    class A objA;
    class B objB;

    int val1, val2;
    cout << "Enter no of for Class A: ";
    cin >> val1;
    cout << "Enter no for Class B: ";
    cin >> val2;
```

```
    objA.setdata (val1);
    objB.setdata (val2);

    findGreatest (objA, objB);

    return 0;
}
```

Output:-

```
Enter number for class A: 30
Enter number for class B:- 45

Greatest number is 45.
```

6] Create two classes, class A & class B, each with a private integer. Write a friend function sum () that can access private data from both classes and return sum.

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

class B; // Forward declaration.

class A {
private:
    int numA;

public:
    A (int a): numA (a) {}

    // Declare friend function
    friend int sum (const A & objA, const B &
```

```
};
class B {
  private:
    int numB;

  public:
    B(int b) : numB(b) {}

    friend int sum(const A &objA, const B &objB);
};

int sum(const A &objA, const B &objB) {
    return objA.numA + objB.numB;
}
int main() {
    A a(55);
    B b(20);

    cout << "sum: " << sum(a,b) << endl;

    return 0;
}
```

Output:-

Sum:- 75

7) Write a program with a class Number that contains a private. Use a friend function Swap Number(Number&, Number&) to swap the private values of two number objects.

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

class Number {
private:
    int value;

public
    // Constructor to initialize the private integer
    Number(int v): value(v) {}

    // Function to display the value
    void display() const {
        cout << value << endl;
    }

    // Declare friend function to access private no
    friend void SwapNumbers(Number &n1, Numbe
};

// Friend function to swap private values of 2 no's ob
void SwapNumbers(Number &n1, Numbe
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp;
}

int main() {
    Number num1(10);
    Number num2(20);

    cout << "Before swaping:" << endl;
    cout << "num1:"
```

```
num1.display();
cout << "num2 = ";
num2.display();

// Swap the private values
swapNumbers(num1, num2);

cout << "After swapping: " << endl;
cout << "num1 = ";
num1.display();
cout << "num2 = ";
num2.display();

return 0;
}
```

Output:-

```
Before swapping:-
num1:-10
num2:-20

After swapping:-
num1:-20
num2:-10
```

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

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

class Cube; // Forward declaration

class Box {
private:
    double volume;

public:
    Box(double length, double width, double height) {
        volume = length * width * height;
    }
    // Declare friend function
    friend void findGreater(Box, Cube);
};

class Cube {
private:
    double volume;

public:
    Cube(double side) {
        volume = side * side * side;
    }

    // Declare friend function
    friend void findGreater(Box, Cube);
};

void findGreater(Box b, Cube c) {
    if (b.volume > c.volume)
```

```
else if (c.volume > b.volume)
    cout << "cube has greater Volume:"
         c.volume << endl;

else if
    cout << "Both have equal volume:"
         << b.volume << endl;
}

int main () {
    Box box (3,4,5); // Volume = 60
    cube cube (4); // Volume = 64

    find Greater (box, cube);

    return 0;
```

Output:-

Cube volume is greater than 64.

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

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

class complex {
private:
    double real;
    double imag;

public:
// constructor
complex (double r = 0, double i = 0): real(r),
    imag(i) {}

// Friend function to add two Complex No's
friend complex add (const complex & c1,
                    const complex & c2);

// Function to display the complex no.
void display () const {
    cout << real;
    if (imag >= 0) cout << " + ";
    cout << imag << " i" << endl;
}
};

// definition of friend function
complex add (const complex & c1, const co
                                        &
    return complex (c1.real + c2.real, c1.im
                                    c2.im

int main () {
    complex c1 (3,4)
    complex c2 (5,-2);

    complex c3 = add (c1, c2);

    cout << "c1 = "; c1.display ();
    cout << "c2 = "; c2.display ();
    cout << "sum = "; c3.display ();
```

```
    return 0;
}
```

Output:-

```
c1 = 3+4i
c2 = 5-2i
Sum = 8+2i
```

10] Create a class student with private data member name & three subject marks. Write a friend function calculate Average (student) that calculates & display the average marks.

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

class student {
private:
    string name;
    int marks1, marks2, marks3;

public:
    student (string n, int m1, int m2, int m3){
        name = n;
        marks1 = m1;
        marks2 = m2;
        marks3 = m3;
    }
    // Declare friend function
    friend void CalculateAverage (student s);
};

// Friend function definition
void calculateAverage (student s) {
    float average = (s.marks1 + s.marks2 + s.marks3) / 3.0;
    cout << "Average marks of " << s.name << " is " << average << endl;
}

int main() {
    student student ("John", 85, 90, 80);
    calculateAverage (student);

    return 0;
}
```

Output:-

Average marks of John is 85.

11] Create three classes: Alpha, Bet & Gamma, each with private data member. Write a single friend function that can access all three and print their sum.

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

class Beta; // Forward declaration
class Gamma; // Forward declaration

class Alpha {
private:
```

```
    int a;
public:
    Alpha (int x) : a(x) {}
    friend void sumAll (Alpha, Beta, Gamma);
};

class Beta {
private:
    int b;
public:
    Beta (int y) : b(y) {}
    friend void sumAll (Alpha, Beta, Gamma);
};

class Gamma {
private:
    int c;
public:
    Gamma (int z) : c(z) {}
    friend void sumAll (Alpha, Beta, Gamma);
};

// Friend function definition
void sumAll (Alpha A, Beta B, Gamma G) {
    int sum = A.a + B.b + G.c;
    cout << "sum of private data members
            is: " << sum << endl;
}

int main () {
    Alpha obj1 (10);
    Beta obj2 (20);
    Gamma obj3 (30);
    sumAll (obj1, obj2, obj3);

    return 0;
}
```

Output:-

Sum of private data members is:- 60

12] Create a class Point with private members x & y. Write a friend function that calcu and returns the distance between two Poin objects.

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

class Point {
private:
    double x, y;

public:
    Point (double xCoord, double yCoord)
        x (xCoord), y (yCoord) {}

    // Declare friend function
    friend double distance (Point P1, Point

};

// Friend function to calculate distance be
    two points
double distance (Point P1, Point P2) {
```

```
    double dx = p1.x - p2.x;
    double dy = p1.y - p2.y;
    return sqrt(dx*dx + dy*dy);
}
int main() {
    Point p1(3.0, 4.0);
    Point p2(7.0, 1.0);

    cout << "Distance between points: "
         << distance(p1, p2) << endl;

    return 0;
}
```

Output:-

Distance between points:-5

13) Create two classes: Bank Account & Audit. Bank Account holds private balance information. Write a friend function in Audit that accesses and prints balance information for Auditing.

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

class Bank; // Forward declaration

class Audit {
public:
    void printBalance(const Bank &b);
};

class Bank {
private:
    double balance;

public:
    Bank(double bal) : balance(bal) {}

    // Declare friend function
    friend void Audit::printBalance(const Bank &b);
};

void Audit::printBalance(const Bank &b) {
    cout << "Auditing Balance: $" << b.balance
         << endl;
}

int main() {
    Bank account(1500.75);
    Audit auditor;

    auditor.printBalance(account);

    return 0;
}
```

Output:-

Auditing Balance: $1500.75

Experiment No: 5

* Write a c++ program to implement types of constructors.

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.

Default constructor:-

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

class sum calculator {
private:
  int n,
  int sum;

public:
  Sum calculator () {
    cout << "Enter the value of n: ";
    cin >> n;
    sum = 0;

    for (int i = 1; i <= n; ++i) {
      sum += i;
    }
  }

void display sum () {
  cout << "The sum of numbers from 1 to " << n << " is : " <<
  sum << endl;
}
};
int main () {
  Sum calculator sc;

  sc.display sum ();
  return 0;
}
```

Output:-

```
Enter the value of n: 45
The sum of numbers from 1 to 45 is:
```

* Parameterised constructor.

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

class Sum calculator {
private:
  int n;
  int sum;

public:
  // Parameterised constructor
  Sum Calculator (int num) {
    n = num;
    Sum = 0;

  // Calculate Sum from 1 to N
  for (int i = 1; i <= n; i++) {
    sum += i;
  }
};
```

```
// Function to display the result
void display Sum() {
    cout << "The sum of numbers from 1 to " << n << " is : " <<
    Sum << endl;
}
};
int main() {
    int number;

    // Take input from the user:
    Cout << "Enter the value of n: ";
    cin >> number;

    // Create object with parameter
    Sum Calculator with SC(Number);

    // Display the sum
    sc.display Sum();

    return 0;
}
```

Output:-

```
Enter the value of n:-67
The sum of no's from 1 to 67 is : 2278
```

Copy constructor:-

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

class Sum calculator {
private:
    int n;
    int sum;

public:
    // Parameterized constructor
    Sum Calculator (int num) {
        n = num;
        Sum = 0;
        for (int i = 1; i <= n; ++i) {
            sum += i;
        }
    }

    // copy constructor
    Sum calculator (const Sum calculator & obj) {
        n = obj.n;
        Sum = obj.sum;
        Cout << "Copy constructor called.";
    }
    // Function to display the result
    void display Sum() {
        Cout << "The sum of no's from 1
            " << n << " is : " << sum
            endl;
    }
};
```

```
int main(){
    int number;

    //Take user input
    cout << "Enter the value of n:";
    cin >> number;

    Sum calculator original(number);
    original.display sum();

    //create a copy using copy constructor
    Sum calculator copy = original;
    copy.display Sum();

    return 0;
}
```

Output:-

```
Enter the value of n:-10
The sum of no's from 1 to 10 is; 55
Copy constructor called.
The Sum of no's from 1 to 10; 55
```

2) Write a program to declare a class "Student" having data members as name & percentage. Write a constructor to initialize these data members. Accept & display data for one student.

1) ~~Write a syntax to define a derived class~~

Default constructor

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

class student {
private:
    string name;
    float percentage;

public:
    // Default constructor
    Student(){
        name = " ";
        percentage = 0.0;
    }

    // Function to accept data
    void accept data(){
        cout << "Enter student name:";
        getline(cin, name);

        cout << "Enter percentage:";
        cin >> percentage;
    }

    // Function to display the data
    void display Data(){
        Cout << "In Student Details:" << end
        Cout << "Name:" << name << endl;
        Cout << "Percentage:" << percentage
             "%." << endl;
```

```
};
int main(){
    // Create Student object
    Student s;

    // Accept & display student data
    s.acceptdata();
    s.displaydata();

    return 0;
}
```

Output:-

Enter student name:- Tejas Katkar
Enter percentage:- 85.5

Student Details

Name:- Tejas Katkar
Percentage:- 85.5 %

* Parameterized constructor

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

class student {
private:
    string name;
    float percentage;

public:
    // Parameterized Constructor
    Student(string n, float p) {
        name = n;
        percentage = p;
    }

    // Function to display student details
    void display() {
        cout << "\n In Student Details
        cout << "Name: " << name << endl;
        cout << "Percentage: " << percentage
    }
};

int main(){
    string name;
    float percentage;

    // Accepting input
    cout << "Enter student name: ";
    getline(cin, name);

    cout << "Enter student percentage:
    cin >> percentage;

    // Creating object using parameter
    Constructor
    Student s(name, percentage);

    // Display student data
    s.display();
    return 0;
}
```

Output:-

Enter Student name:- Alice Johnson
Enter Student Percentage:-87.5
Name:- Alice Johnson
Percentage:- 87.5%

* Copy Constructor

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

class student {
    string name;
    float percentage;

public:
    // Parameterized Constructor
    Student (string n, float p) {
        name = n;
        percentage = p;
    }
    // Copy constructor
    Student (const Student &s) {
        name = s.name;
        percentage = s.percentage;
    }
    void display () {
        cout << "Name: " << name << endl;
        cout << "Percentage: " << percentage
             << "%" << endl;
    }
};
int main () {
    string name;
    float percentage;

    cout << "Enter Student name
    getline (cin, name);
    cout << "Enter student percentage:"
    cin >> percentage;

    // Create first student object
    Student s1 (name, percentage);

    // Create second student object using
       copy constructor
    Student s2 = s1;

    cout << "\n Data of copied Student:" <<
    s2.display ();

    return 0;
}
```

Output:-

Enter student name: John
Enter Student percentage: -92.3

Data of first student:
Name: John
percentage :- 92.3%

c) Define a class 'College' membess 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 & display the data.

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

class college {
    int roll_no;
    string name;
    string course;

public:
    // Constructor with default value for course
    college (int r, string n, string c = "Computer engineering")
    {
        roll_no = r;
        name = n;
        course = c;
    }

    void display () {
        cout << "Roll No: " << roll_no << "\n";
        cout << "Name: " << name << "\n";
        cout << "Course: " << course << "\n";
    }
};

int main () {
    int roll_no;
    string name, course;

    // Input for first object
    cout << "Enter details for student 1: \n";
    cout << "Roll no: ";
    cin >> roll_no;
    cin.ignore (); // Ignore newline character left in buffer

    cout << "Name: ";
    getline (cin, name);
    cout << "course"
    getline (cin, course)

    College student1 (roll_no, name, course.empty
                ? "computer engineering" :

    // Input for second object
    cout << "\n Enter details for student 2: \n
    cout << "Roll No: ";
    cin >> roll_no;
    cin.ignore ();
    cout << "Name: ";
    getline (cin, name);
    cout << "course (press enter to accept
                default - computer eng
    getline (cin, course);

    college student2 (roll_no, name, course
                empty ()? "compeng

    // Display details
    cout << "\n Student 1 details: \n";
    Student1.display ();
```

```
    cout << "\n Student 2 Details : \n";
    Student2.display();

    return 0;
}
```

Output:

```
Enter details for student1:
Roll No: 45
Name: Arjun Kadam

Enter details for Student 2:
Roll No:-67
Name:-Tejas Katkar

Student1 Details:
Roll No: 45
Name:-Arjun Kadam
Course:- Computer Engineering

Student 2 details:-
Roll No:-67
Name:- Tejas Katkar
Course:- Computer Engineering.
```

Write a program to demonstrate constructor overloading.

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

class Box {
private:
    double length;
    double breadth;
    double height;

public:
    // Default constructor
    Box() {
        length = 0;
        breadth = 0;
        height = 0;
        cout << "Default constructor called."
             << endl;
    }

    // constructor with one parameter
    Box(double l) {
        length = l;
        breadth = l;
        height = l;
        cout << "Single parameter constr
             called." << endl;
    }

    // Constructor with three parameters
    Box(double l, double b, double h) {
        length = l;
        breadth = b;
        height = h;
        cout << "Three parameter constructor called." <<
```

```
// Function to display volume
void displayVolume() {
    double volume = length * breadth * height;
    cout << "Volume: " << volume << endl;
}
};

int main() {
    // Calling default constructor
    Box box1;
    box1.displayVolume();

    // Calling single parameter constructor
    Box box2(5);
    box2.displayVolume();

    // Calling three parameter constructor
    Box box3(3, 4, 5);
    box3.displayVolume();

    return 0;
}
```

Output:-

```
Default constructor called.
Volume: 0
Single parameter constructor called.
Volume: 125
Three parameter constructor called.
Volume: 60.
```

15/10

Experiment No:-6

1.) Single Inheritance

Problem:- Create a base class called Person with attributes name & age. Derive a class Student from Person that adds an attribute rollNumber. Write functions to display all details of the Student.

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

// Base class
class Person {
protected:
    string name;
    int age;

public:
    // constructor
    Person(string n, int a) {
        name = n;
        age = a;
    }
};

// Derived class
class Student : public Person {
private:
    string rollNumber;
```

```
public:
    // Constructor
    Student(string n, int a, string r) : Person(n, a)
        rollNumber = r;
}
// Function to display student details
void displayDetails() {
    cout << "Student Details:" << endl;
    cout << "Name : " << name << endl;
    cout << "Age : " << age << endl;
    cout << "Roll Number: " << rollNo << endl;
}
};
// main function
int main() {
    Student student1("Alice", 20, "S12345");
    student1.displayDetails();

    return 0;
}
```

Output

Student Details:

Name :- Alice
Age :- 20
Roll Number: S12345

2) multiple Inheritance

Problem :- Create two base classes Academic Sports

* Academic class contains marks of a studen
* Sports class contains marks of a student
Create a derived class Result that inher from both Academic & Sports.

Write a function to calculate total score & display details.

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

// Base class 1
class Academic {
protected:
    float academicMarks;

public:
    void setAcademicMarks(float marks) {
        academicMarks = marks;
    }
};

// Base class 2
class Sports {
protected:
    float sportsScore;

public:
    void setSports(float score) {
```

```
        SportsScore = score;
    }
};

// Derived class using Multiple Inheritance
class Result: public Academic, public Sports {
private:
    string studentName;

public:
    void setDetails(string name, int mks, int score) {
        studentName = name;
        setAcademicMarks(marks);
        setSportsScore(score);
    }
    int calculateTotal() {
        return academicMarks + sportsScore;
    }
    void displayDetails() {
        cout << "Student Name : " << studentName << endl;
        cout << "Academic Marks : " << academicMarks() << endl;
        cout << "Sports Score : " << sportsScore() << endl;
        cout << "Total Score : " << calculateTotal() << endl;
    }
};

// main function
int main() {
    Result student;

    string name;
    int academicMarks, sportsScore;

    // Input
    cout << "Enter Student Name : ";
    getline(cin, name);
    cout << "Enter academic marks (out of 100) : ";
    cin >> academicMarks;
    cout << "Enter sports score (out of 50) : ";
    cin >> sportsScore;

    // Set & display
    student.setDetails(name, academicMarks, sportsScore);

    cout << "\n--- Student Result ---\n";
    student.displayDetails();

    return 0;
}
```

Output:-

```
Enter student name:- Alice
Enter Academic marks (out of 100) : - 85
Enter sports score (out of 50) : - 40

--- Student Result ---
Student Name : - Alice
Academic Marks : - 85
Sports Score : - 40
Total Score :- 125
```

3) Multilevel Inheritance

Problem -
Create a class Vehicle with attributes like brand & model.
Derive two classes manager & Developer from employee.
manager has an attribute department & developer has an attribute programming Language.
Write functions to display details for both.

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

// Baseclass : Vehicle
class Vehicle {
protected:
    string brand;
    string model;

public:
    void setVehicleDetails(const string& b,
                           const string& m) {
        brand = b;
        model = m;
    }
};

// Derived class: carC inherits from Vehicle
class Car : public Vehicle {
protected:
    string type; // e.g., Sedan, SUV

public:
    void setCarType(const string& t) {
        type = t;
    }

// Derived class : Electric car C inherits from car
class ElectricCar : public Car {
private:
    int batteryCapacity; // in kWh

public:
    void setBatteryCapacity(int capacity) {
        batteryCapacity = capacity;
    }

    void displayDetails() {
        cout << "\n---- Electric Car Details ----" << endl;
        cout << "Model    : " << model << endl;
        cout << "Type     : " << type << endl;
        cout << "Battery Capacity: " << batteryCapacity << " kWh"
             << endl;
    }
};

// main function
int main() {
    ElectricCar myEV;

    string brand, model, type;
    int battery;

    // Input from user
    cout << "Enter vehicle brand: ";
    getline(cin, brand);
    cout << "Enter vehicle model: ";
    getline(cin, model);
    cout << "Enter car type (eg, Sedan, SUV): ";
    getline(cin, type);
```

```
cout << "Enter battery capacity (in kWH):";
cin >> battery;

// Set details
myEV.setVehicleDetails(brand, model);
myEV.setCarType(type);
myEV.setBatteryCapacity(battery);

// Display details
myEV.displayDetails();

return 0;
}
```

Output:-

```
Enter vehicle brand:- Tesla
Enter vehicle model:- Models
Enter car type (e.g, Sedan, SUV): Sedan
Enter battery capacity (in kWH): 100

Electric Car Details:-

Brand    : Tesla
Model    : Models
Type     : Sedan
Battery capacity: - 100 kWH
```

4) Hierarchical Inheritance

Problem:-
Create a class vehicle with attributes like brand &m...

Derive a class manager & Developer from employee; manager has an attribute department & developer has an attribute programming language. write functions to display details for both.

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

// Base class
class Employee {
protected:
    int empID;
    string name;

public:
    Employee(int id, string n) {
        empID = id;
        name = n;
    }
    virtual void display() {
        cout << "Employee ID: " << empID << endl;
        cout << "Name: " << name << endl;
    }
};

// Derived class: manager
class manager: public Employee {
private:
    string department;

public:
    manager(int id, string n, string dept):
        Employee(id, n) {
```

```
    department = dept;
}
void display() override {
    Employee::display();
    cout << "Department: " << department << endl;
}
};

// Derived class: Developer
class Developer: public Employee {
private:
    string programmingLanguage;

public:
    Developer(int id, string n, string lang): Employee(id, n) {
        programmingLanguage = lang;
    }
    void display() override {
        Employee::display();
        cout << "Programming Language: " << programmingLanguage << endl;
    }
};

int main() {
    Manager m1(101, "Alice", "HR");
    Developer d1(102, "Bob", "C++");

    cout << "Manager Details:" << endl;
    m1.display();

    cout << "\nDeveloper Details:" << endl;
    d1.display();

    return 0;
}
```

Output:-

```
Manager Details:
Employee ID: 101
Name: Alice
Department: HR

Developer Details
Employee ID: 102
Name: Bob
Programming Language: C++
```

5) Hybrid Inheritance

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

// Base class Person
class Person {
protected:
    string name;
    int age;

public:
    Person(string n, int a): name(n), age(a) {}
    void display() {
        cout << "Name: " << name << endl;
```

```
// Student class derived from Person (multilevel)
class Student : public Person {
protected:
    int marks[3]; // Assume 3 subjects

public:
    Student(string n, int a, int m1, m2, m3):
        Person(n, a) {
        marks[0] = m1;
        marks[1] = m2;
        marks[2] = m3;
    }
    int getTotalMarks() {
        return marks[0] + marks[1] + marks[2];
    }
    void displayMarks() {
        cout << "Marks: " << marks[0] << ", "
             << marks[1] << ", " << marks[2] << endl;
    }
};

// Sports class
class Sports {
protected:
    int sportsScore;

public:
    Sports(int score): SportsScore(score) {}
    int getSportsScore() {
        return sportsScore;
    }
    void displaySportsScore() {
        cout << "Sports Score: " << sportsScore << endl;
    }
};

class Result: public Student, public Sports {
public:
    Result(string n, int a, int m1, m2, m3, int score)
        : Student(n, a, m1, m2, m3), Sports(score) {}
    void displayResult() {
        displayPerson();
        displayMarks();
        displaySportsScore();
        cout << "Total Marks: " << getTotalMarks() << endl;
        cout << "Total Score (Marks + Sports): " << getTotalMarks() + getSportsScore() << endl;
    }
};

int main() {
    Result r1("John", 20, 85, 90, 88, 50);
    cout << "Student Details:" << endl;
    r1.displayResult();
    return 0;
}
```

Output:-

Result Details:-

Name:-John
Age:-20
marks:-85, 90, 88
sports score: 50
Total mks:-263
TotalScore (Mks+Sports): 313.

16/10

Experiment No:-7

Write a program to demonstrate the Compile Time Polymorphism (Function Overloading & operator Overloading-Unary).

a) Write a program using function overloading to calculate area of a laboratory

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

//Function to calculate area of a rectangle
double area (double length, double width) {
    return length * width;
}
//Function to calculate area of square
double area (double side) {
    return side * side;
}
int main() {
    double length, width, side;

//Input for library (Rectangle)
cout << "Enter side of a the classroom:";
cin >> side;
cout << "Area of the classroom (square): "
    << area(side) << endl;

return 0;
}
```

Name:-John
Age:-20
marks:-85, 90, 88
sports Score:-50
Total MKS:-263
TotalScore (MKS+Sports):-313.

16/10

Experiment No:-7

Write a program to demonstrate the Compile Time Polymorphism (Function Overloading & operator Overloading-Unary).

a) Write a program using function overloading to calculate area of a laboratory

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

//Function to calculate area of a rectangle
double area (double length, double width){
    return length * width;
}
//Function to calculate area of square
double area (double side) {
    return side * side;
}
int main(){
    double length, width, side;

//Input for library (rectangle)
cout << "Enter side of a the classroom:";
cin >> side;
cout << "Area of the classroom (square):
    << area(side) << endl;

return 0;
}
```

Enter length & width of library :- 20 15
Area of the library (rectangle) : 300
Enter side of classroom : 10
Area of ~~eta~~ classroom (library) : 100

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

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

// Function to sum 5 float values
float sum (float a, float b, float c, float d, float e){
    return a+b+c+d+e;
}

// Function to sum 10 integer values
int sum (int a1, int a2, int a3, int a4, int a5, int a6,
         int a8, int a9, int a10) {
    return a1+a2+a3+a4+a5+a6+a7+a8+a9+a10;
}

int main () {
    // Sum of 5 float values
    float f1=1.1, f2=2.2, f3=3.3, f4=4.4, f5=5.5;
    float floatSum=sum (f1, f2, f3, f4, f5);
    cout << "Sum of 5 float values: " << floatSum
         << endl;

    // Sum of 10 integer values
    int i1 = 1, i2 = 2, i3=3, i4=4, i5=5, i6=6,
        i7=7, i8=8, i9=9, i10=10;
    int sum=sum (i1, i2, i3, i4, i5, i6, i7, i8, i9, i10);
    cout << "Sum of 10 integer values: " << sum
         << endl;

    return 0;
}
```

output :-

Sum of 5 float values : 16.5
Sum of 10 integer values : 55

c) Write a program to implement unary operator, when used with the ~~used~~ object to that the numeric data of the class is negated.

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

Class Number {
private:
    int a, b

public:
    // Constructor
    Number (int x, int y) {
        a=x;
        b=y;
    }
    // overload unary minus (-) operator
    Void operator -() {
        a = -a;
        b = -b;
```

```
};
// Display function
void display() {
    cout<<"a = "<<a<<", b = "<<b<<endl;
}
};

int main() {
    Number num(10, -20);

    cout << "Before negation: " << endl;
    num.display();

    // Apply unary operator
    -num;

    cout << "After negation: " << endl;
    num.display();

    return 0;
}
```

Output:-

Before negation:-
a = 10, b = 20.

After negation:-
a = -10 b = -20

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

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

class Number {
private:
    int value;

public:
    // Constructor
    Number(int val) {
        value = val;
    }
    // Pre-increment : ++obj
    Number& operator++() {
        ++value;
        return *this;
    }
    // Post-increment : obj++
    Number operator++(int) {
        Number temp = *this;
        value++;
        return temp;
    }

    // Display function
    void display() {
        cout<< "Value = " << value << en
    }
};
```

```
int main () {
    Number num (5);

    cout << "original : ";
    num. display ();

    cout << "\n After Pre-Increment (++num): ";
    ++num;

    num. display ();

    cout << "\n After Post-Increment (num++): ";
    num++;
    num. display ();

    return 0;
}
```

output:-

original value :- 5

Pre - Increment (++num): Value = 6

Post - Increment (num++): Value = 7

27/10

Experiment No:- 8

1) WAP to overload '+' operator so that two string can be concatenate eg: "xyz" + "par" = xyzpar.

```
#include <iostream>
#include <string>
using namespace std;
class Mystring {
public: string str;
My string operator+ (My string& other) {
temp str = str + other str;
return temp;
}
void display ()
{
cout << str << endl;
}
};
int main ()
{
My string s1, s2, s3;
s1.str = "xyz";
s2.str = "par";
s3 = s1 + s2;

cout << " concarerated string;
s3.display ();
return 0;
}
```

2) WAP to create base class login having data members as name & password. Declare accept() function virtual. Derive emaillogin & membership login.

```
#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 emailogin : 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 membership login: Public login {
    String membership id;
  public:
    void accept() override {
      login::accept();
      cout<<"Enter membership id";
      cin>>membership id;
    }
    void display() {
      cout<<"\n -- Membership login be----
      cout<<"Name"<<name<<"\n password
      password<<"membership id"<<
    }
};
int main()
{
  email login e;
  Membership login m;
  e.accept();
  m.accept();
  e.display();
  m.display();
  return 0;
}
```

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.txt" in read (ios::in) mode & "Second.txt" file in write (ios::out) mode. Copy the contents of "First.txt" into "Second.txt". Assume "First.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 file stream (write mode)
  ofstream outFile("second.txt", ios::out);

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

```
  inFile.close();
  return 1;
}
  char ch;
  // Read character by character from First.txt &
     write to Second.txt
  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." <<
    return 1;
}
char ch;
int digitCount = 0;
int spaceCount = 0;

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

file.close();

cout << "Number of digits: " << digitCount << endl;
cout << "Number of spaces: " << spaceCount << endl;

return 0;
}
```

Q3) Write a C++ to 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 << "." << endl;
        return 1;
    }

    string word;
    int wordCount = 0;

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

    file.close(); // Close the file

    cout << "Total no of words in the file
        << wordCount << endl;

    return 0;
}
```

Q9. Write a C++ program to count digits & spaces using file handling.

```
#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: " << spaceCount << endl;
    return 0;
}
```

Q10. Write a C++ program to count occurrence of a given word using File Handling.

```
#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;

    while (fin >> word) {
        if (word == search)
            count++;
    }

    cout << "Occurrence of \"" << search << "\": "
         << count << endl;

    fin.close();
    return 0;
}
```

Experiment No:-10

* Write a program to implement a Function template & class template.

a) Write a c++ program to find Sum of array elements using function template. Ceg. Pass Integer, Float, Double array of 10 elements)

```
#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 sum;
}

int main () {
    int a[5] = {1,2,3,4,5}
    float b[5] = {1.2, 2.3, 3.4, 4.5, 5.6};
    double c[5] = {1.11, 2.22, 3.33, 4.44, 5.55};

    cout << "Sum of Integer Array: " << sumArray(a,5) << endl;
    cout << "Sum of Float Array: " << sumArray(b,5) << endl;
    cout << "Sum of Double Array: " sumArray (c,5) << endl;
    return 0;
}
```

b) Write a c++ program of Square Function using template Specialization.

- Calculate the square of integer no. and a String. C square of string is nothing but Concantenation of a string with itself().
- Write a specialized function for the square of a string.

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

template (class T)
T square (T x) {
    return x = x;
}

// Specialization for string
template <>
String square (string s) {
    return s+s; // string concatenation with itself
}

int main () {
    int a = 5;
    String str = "ABC";

    cout << "square of integer: " << square(a) << endl;
    cout << "Square of String: " << square(str) << endl;

    return 0;
}
```

c) Write a c++ program to build Simple Calculator using a class template.

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

template <class T>
class calculator {
   T a, b;
public:
   calculator(T x, T y) {
        a = x;
        b = y;
}

void add() { cout << "Addition:" << a+b << endl; }
void sub() { cout << "Subtraction:" << a-b << endl; }
void mul() { cout << "multiplication:" << a*b << endl; }
void div() { cout << "Division:" << a/b << endl; }
};
int main() {
   Calculator <int> obj1(10,5);
   Calculator <float> obj2(5.5, 2.2);

   cout << "--- Integer Calculator ---" << endl;
   obj1.add();
   obj1.sub();
   obj1.mul();
   obj1.div();

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

   return 0;
}
```

d) Write a c++ program to implement Generic ~~vector~~. Include the following ~~fn~~ push-pop methods from stack using class template.

```
#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 == 19)
         cout << "Stack overflow!" << endl;
      else
         s[++top] = x;
   }

   void pop() {
      if (top == -1)
         cout << "Stack underflow!" << endl;
      else
         cout << "Popped: " << s[top--] << endl;
   }
```

```
void display (){
    if(top == -1)
        cout << "Stack is empty!" << endl;
    else {
        cout << "Stack is Empty!" << endl;
        for (int i = top; i >= 0; i--)
            cout << s[i] << " ";
        cout << endl;
    }
};

int main (){
    Stack <int> St;
    st.push(10);
    st.push(20);
    st.push(30);
    st.display();
    st.pop();
    st.display();

    return 0;
}
```

Experiment No:-11

* Write a C++ program to implement generic vectors. Include following member function To create the vector.
a) To modify the value of a given element
b) To multiply by a scalar value
c) To display the vector in the form (10,20,30...)

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

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

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

    // create from existing vector
    GenericVector (const std::vector <T> &v)
    {}

    // To create/reset vector (provide new elemen
    void create (const std::vector <T> &v)
        data = v;
}
```

```
// Modify the value of a given element (0-based index)
void modify(size_t index, const T& value) {
    if (index >= data.size())
        throw std::out_of_range("Index out of range in modify()");
    data[index] = value;
}
// Multiply each element by a scalar (same type)
template <typename>
void multiplyByScalar(const U& scalar) {
    for (auto& elem : data) {
        elem = static_cast<T>(scalar);
    }
}

// Display in form (a, b, c, ....)
void display() const {
    cout << "(";
    for (size_t i = 0; i < data.size(); ++i) {
        cout << data[i];
        if (i + 1 < data.size()) cout << ", ";
    }
    cout << ")";
}

// Utility: size
size_t size() const { return data.size(); }

// Utility: get element (const)
const T& at(size_t i) const {
    if (i >= data.size())
        throw std::out_of_range("Index out of range");
    return data[i];
}
};

int main() {
    // Create int vector
    GenericVector<int> gv = {10, 20, 30, 40};
    cout << "Initial int vector: ";
    gv.display();
    cout << "\n";

    // Modify element
    gv.modify(2, 300);
    cout << "After modify index 2 -> 300: ";
    gv.display();
    cout << "\n";

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

    // Create double vector
    GenericVector<double> gv2 = {1.5, 2.5, 3.5};
    cout << "Double vector: ";
    gv2.display();
    cout << "\n";
    gv2.multiplyByScalar(3.0);
    cout << "After multiply by 3.0: ";
    gv2.display();
    cout << "\n";

} catch (const exception& e) {
    cerr << "Error: " << e.what() << "\n";
    return 1;
}
```

Experiment No:-12.

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

int main(){
  stack<int> st;

  // Push elements
  st.push(10);
  st.push(20);
  st.push(30);

  cout<<"Stack contents (top to bottom):";
  // To display without destroying original copy
  stack<int> temp=st;
  while(!temp.empty()){
    cout<<temp.top()<<" ";
    temp.pop();
  }
  cout<<"\n";

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

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

b)

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

int main(){
  queue<int> q;
  q.push(10);
  q.push(20);
  q.push(30);
  cout<<"front:"<<q.front()<<endl;
  q.pop();
  cout<<"front after pop:"<<q.front()<<
      endl;
  return 0;
}
```

Personal Information

tejas.katkar@mitwpu.edu.in

Add your mobile number

India

My Resume

+ Add Resume

Add your resume here

EEO settings

My Badges

My Certifications

You have not earned any certificates yet. Get Certified

Work Experience

Add your work experience. Don't forget to add those internships as well.

Education

We believe in skills over pedigree: but go ahead add your education for the recruiters who don't.