Problem Statement :-

① Write the C++ program for addition of two numbers :

→
```
#include <iostream.h>
int main () {

int a, b, c;
Cout << "Enter val of a & b";
Cin >> a >> b;
 c = a+b;
Cout << "Addition = " << c;
}
return 0;
}
```

Output

Enter val of a & b: 5 3
Addition: 8.

② Write the C++ program for arithmetic operations using switch case.

→
```
#include<iostream.h>
int main(){
 char op;
float num1, num2;
cout<<"Enter operation (+,-,*,/):";
cin>>num1>>num2;
    switch (op){
    case '+':
cout<<"Result:"<<num1+num2;
    break;
    case '-':
cout<<"Result:"<<num1-num2;
    break;
    case '*':
cout<<"Result:"<<num1*num2;
    break;
    case '/':
    if(num2;=0)
    cout<<"Result:"<<num1/num2;
else
 cout<<"Division by in not allowed";
  break;
default:
cout<<"Invalid operator";
}
return 0;
}
```

③ Write a C++ program to check if the number is even or odd.

→
```
#include<iostream.h>
  int main(){
   int num;
  cout<<"Enter a number for checking it;
          it is even or odd:";
   (int>>num;

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

④ Write a C++ program to print 1 to 10 numbers using a for loop.

→
```
#include <iostream.h>
int main () {
for ( int i=1;i<=10;i++) {
cout <<i<< " ";
}
return 0;
}
```

⑤ C++ program to print 10 to 1 numbers using a while loop.

→
```
#include <iostream.h>
int main () {
int i = 10;
while (i >=1) {
cout <<i<< " ";
i--;
}
return 0;
}
```

⑥ Pattern :-

(i)
```
*
* *
* * *
```

→
```
#include <iostream.h>
using namespace std;
void main () {
cout << "Pattern a: Star triangle";
int n=3; k=n;
for( int i=1;i<=n;j++) {
if (j>=k)
cout << "*";
else cout << " ";
} cout <<endl; }
```

(ii)
```
1
22
333
4444
55555
```

→
```
cout << " Pattern b: Number triangle 1";
for (int i=1, i<=5 ;i++){
```

```
for(int j<=1;j++){
  cout<<i<<" ";
} cout<<endl;}
```

(iii)
```
1
12
123
1234
12345
```

→
```
cout<<" Pattern C: Number pattern 2";
for(int i=1;i<=5;i++){
  for(int j=1;j<=i;j++){
    cout<<j<<" ";
  } cout<<endl;}
}
```

15/7/25

Experiment -1

Problem statement :-

Write a C++ program to declare a class Student having data member as roll_no and name. Accept and display data for single student.

Code :-

```
#include<iostream.h>
  using namespace std;
  class Student{
    int roll_no;
    String_name;
    public :

  void get_data(){
    Cout<<"Enter student name and roll no";
    cin>>name>>roll_no;
  }
    void display(){
    Cout<<"Name:"<<name;
    Cout<<"Roll no"<<roll_no;
    }
};
```

```
int main (){
Student S1;
S1.get_data ();
S2.display ();
return 0;
}
```

→

② Problem Statement :
Write a cprogram to create a class book having data members as bname, bprice, bpages. Accept the data for two book and display the name of book having greater price :-

#code :-

```
#include <iostream.h>
using namespace std;
class book {
  public: int bprice, bpages;
  public: int string bname;
  public: void accept (){

    cout << " Enter name, price and pages: ");
    cin >> bname >> bprice >> bpages;
  }
};

int main (){
  book b1, b2;
  b1.accept ();
  b2.accept ();
  cout << "Book with greater price:"
    << (b1.price > b2.price? b1.name : b2.name);
```

}

~~#output~~

③ Problem Statement :- [15-7-25]

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

code :

```
#include <iostream>
using namespace std;
class time1 {
  int h,m,s;
  public:
  void accept (){

cout << "Enter time in HH:MM:SS format"
                    endl;
cout << "Hours (0-23): ";
  cin >> h;
  cout << "Minutes (0-59): ";
  cin>> m;
  cout << "Seconds (0-59): ";
   cin >> s;
}
  void display () {
  cout << "\n Entered time: " << h << ":" << m <<
                    ":" << s;
  cout << "\n Total Seconds:" << (long int) h * 3600 +
          (long int) m * 60 + (long int) s) << endl;
}
};

int main () {
  time1 t1;
  t1.accept ();
  t1.display ();

  return 0;
}
```

15/7

Experiment - 2

1. Write a C++ program to demonstrate the use of class & Object:

(a) 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.

Code :-

```
#include <iostream>
#include <string>
#include <vector>

Class City
{
public:
std::string name;
int population

void accept()
{
std:cout << "City name: " << name << std::end
std::cin >> name >> population
}
void display()
{
    std:: cout << "City name:" << name << std:: end;
    Std:: cout << "Population:" << population << std::endl;
}
};

int main()
{
    std:: vector <City> cities(5);
    for (int i = 0; i<5; i++)
{
    Std:: cout << "\n Enter data for city" << i+1 << ":" <<
                    std::endl;
        cities [i]. accept();
    }

    City highestpopulaityCity = cities[0];
    for (int =1 ; i<5; i++)
{
        if (cities [i]. population > highestpopulationCity.
                                        population)
        {
```

highestPopulationCity = cities[i];

}

}

std::cout << "\n City with highest Popula
<< std::endl;

std::cout << "Name: " << highestPopulatio
name << std::endl;

std::cout << "Population: " << highestPopulat
population << std::endl;

return 0;

}

Output

Enter data for City 1:
Enter name and population of the city: Delhi 100

Enter data for City 2:
Enter name and population of the city: Mumbai

Enter data for City 3:
Enter name and population of the city: KolKata 90

Enter data for City 4:
Enter name and population of the city: Banglore 1003

Enter data for City 5:
Enter name and population of the city: Chennai 10000

City with the highest population
Name: Chennai
Population: 10000

(b) Write a program to declare a class "Account" having data member as Account no. and balance. Accept this data for 10 accounts & give interest 10% where balance is equal or greater than 5000 & display them:

Ans — # Code

```
#include <iostream>
#include <string>
#include <vector>

Class Account
{
  Public;
   std:: String name;
```

long long acc...
double balance;

Experiment -3

a) Write a program to declare a class "Book" containing data membess as book-title, author-() name and price. Accept & display the Information for one object using pointer in that object.

Ans →

```
#include <iostream>
#include <string>
#include <limits>
  Using namespace Std;

Class Book {
private:
  String title;
  String author;
  Double price;
public:
  void acceptInfo () {
  Cout << " Enter Book title: ";
  Cin.ignore (numeric_limits<streamsize > max(), '\n')
  getline (cin, title);
```

```
getline (cin, title);
Cout << " Enter Price: ";
  Cin >> price;
}
void displayInfo () {

Cout << "\n --- Book Details ---\n";
Cout << "Title" << title << "\n";
Cout << "Author" << author << "\n";
Cout << "Price" <<
```

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

⇒

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

class student {

  Char name;
   int roll_no;
   .
  Public:
void accept () {
Cout << "Write it please";
 Cin >> this → name → this → roll → this;
}
  void display () {
  this → accept ()
Cout << this → name → this → rollno << ;
}
```

```
};
int main (){

Student S1
S1.display ();
return 0;
}
```

c) WAP to demonstrate the use of nested class.

~~Code:-~~

~~#include <stdio.h~~

Code:-

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

Class Student {
  private:
    String name;

public:
    Class Address {
  private:
    String city;
    String country;
  Public:
  Address(const string& city = " ", const string&
      country = " ")
    : city(city), country(country) {}
```

```
void accept_address (){
    cout << "Enter City: ";
    getline (cin, city);
    cout << "Enter Country: ";
    getline (cin, city); country;)
}
void display_address () const {
    cout << "Address: " << city << ", " <<
        country << endl;
    }
};
Address Student_address;
Student(): name (" "), Student_address
(" ", " ") { }

void accept_data (){
    cout << "--- Enter Student data---
        endl;
    cout << "Enter student Name: ";
    getline (cin, name);

    Student_address.accept_address ();
}
void display_data () const {
    cout << "\n--- Displaying Student Data ---"
        << endl;
    cout << "Student Name: " << this->name << endl;

    this -> student_address.display_address ();
}
};

int main (){
    Student student;
    Student.accept_data ();
    Student.display_data ();
    Cout << "\n" --- Creating a staudalone
        Address object --- " << endl;
    Student :: Address new_address ("London",
        "UK");
    new.address.display_address ();

    return 0;
}
```

12/8/25

Experiment – 4

Q1. Write a C++ program to resolve the following problems statements

1) Swap 2 numbers from same class using objects as functions arguments. Write swap func. as members function.

Code :-

```
#include <iostream.h>
using namespace std;
class num {
public:
  int n1, n2;
  void accept () {
     cout << " Enter first no. : " <<endl;
     cin >> n1;
     cout << " Enter Second no: " <<
     cin >> n2;
  }
  void display () {
     cout << " Num1 : " << n1 << endl;
     cout << " Num2 : " << n2 <<
  } void swap (num) {
     int temp = n1 = n2; n2 = temp;
     cout << " Swapped : \n First no. : " << n1 << \n
          Second no. " << n2;
     }
  } nu;
  int main () {
    nu. accept ();
    nu. display ();
    nu. swap ( nu );
    return 0;
  }
```

O/P :-

```
Enter first no:
   5
Enter Second no:
   9
num1 : 5
num2 : 9
Swapped:
 num1 : 9
 num2 : 5.
```

2. Swap two numbers from same class using concept of friend function.

Code:-

```
#include <iostream.h>
using namespace std;
class num{
public:
    int n1, n2;
    void accept () {
    cout << "Enter first number:"<<
    cin >> n1;
    Cout << " Enter Second no.": <<
    cin >> n2;
}
void display () {
    cout << "Num1 : " << n1 << endl;
        "Num 2": << n2 << endl;
}
};
void Swap (num) {
    int temp = nu.n1;
    nu.n1 = nu.n2;
    nu.n2 = temp;
    Cout << " Swapped: \n First:
        "<< nu.n1 << " \n Second: "no.n2 <<
                endl;
}
int main () {
    nu.accept ();
    nu.display ();
    Swap (nu)
    return 0;
}
```

O/P:-

```
Enter first no:
    5
Enter Second no.
    9
num1 : 5
num 2 : 9
  First: 9
  Second: 5.
```

3. Swap two numbers from different class using friend function.

Code :-

```
#include <iostream.h>
Using namespace std;
class num1 {
public: int n1;
    void accept () {
    cout << "Enter first no:";
    cin >> n1; }
void display (){ cout << "Num1: " <<
                    endl; }

}nu1;
class num2 {
public : int n2;
    void accept () {
    cout << "Enter second no: "<<
    cin >> n2 ;}
void display (){ cout << "Num2: "<<
                    endl; }
}nu2;
void swap (num1, num2) {
    int temp = nu1.n1;
    nu1.n1 = nu2.n2;
    nu2.n2 = temp;
cout << "Swapped: \n First: " << nu1.n1 << \n
    Second: " << nu2.n2 << endl;
}
int main () {
    nu1.accept();
    nu2.accept ();
    nu1.display();
    nu2.display();
    swap (nu1, nu2);
    return 0;
}
```

O/P

```
Enter first no: 5
Enter Second no: 9
Num1: 5
num2: 9
  Swapped:
    First: 9
    Second: 5.
```

4. Create 2 classes Result1 and Result2 which stores the marks of the student. Read the value of marks for both the class objects & compute avg.

Code:-

```
#include <iostream.h>
using namespace std;
Class Result {
public:
  int r1;
  void accept (){
    cout << "Enter first result: ";
    cin>> r1;}
}res1;
Class Result2 {
public:
  int r2;
  void accept () {
    Cout << " Enter second result: ";
    cin >> r2;}
}res2;
float avg(result1, Result2){
    return (res1.r1 + res2.r2)/2;
}
int main (){
  res1.accept ();
  res2.accept ();
float avg = avg (res1, res2);
  cout << "Average: " << avg;
    return 0;
  }
```

O/P:

Enter first result: 95
Enter Second result: 99.

Avg: 97.

5. Find the greatest number among 2 numbers from 2 different classes using friend function.

Code:-

```
#include <iostream.h>
using namespace std;
Class num1{
public: int n1;
    void accept (){
```

```
    cout << "Enter first number: " << endl;
    cin >> n1; }
} n1;
class num2 {
public: int n2;
  void accept (){
    cout << " Enter Second number: ";
    cin >> n2; }
} n2;
int gnum(int n1, int n2) { return (n1 > n2
                                   n1 : n2
int main (){
  n1.accept ();
  n2.accept ();
int gr = gnum (n1.n1; n2.n2);
cout << "Greatest : " << gr << endl;
return 0;
}
```

O/P :-

Enter first no: 5
Enter Second no: 9
Greatest : 9.

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

Code:

```
#include <iostream.h>
    using namespace std;
    class classB;
    class classA;
    int A;
  public;
    void accept (){
    cout << "Enter value for A: ";
      cin >> A;
  }
    friend int Sum (classA oA, classB oB);
} na;
  class classB{
      int B;
  public;
    void accept (){
      cout << "Enter value for B: ";
        cin >> B;
  }
```

friend int sum(
};nb;
int sum (ClassA OA, class B OB) {
return OA.A + OB.B;
}
int main (){
na.accept ();
nb.accept ();
cout << "Sum:" << sum(na,nb) <<
return 0;
}

O/P :-

Enter Value for A: 3
Enter Value for B: 6
Sum: 9.

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

Code:

```
#include <iostream.h>
Using namespace std;
Class Number {
    int num;
Public;
    void accept () {
    cout << "Enter value: ";
    cin >> num;
    }
    void display () {
    cout << "Value": < num << end;
    }
    friend void swap numbers (Number & num1
        Number &, num2);
} n1, n2;
void swapnumbers (Number & , number &
    int temp = n1.num;
    n1.num = n2.num;
    n2.num = temp;
```

```
cout << "Enter side length for the cube:
cin>>S;
V = S * S * S;
} friend void findGreater (Box B, Cube C);
};
void findGreater (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;
}
```

O/P:

Enter l, w, h for the Box: 3, 4, 5.
Enter side length for the cube: 3.
Greater volume : Box.

4. Create a class Complex with real and imaginary parts as a private member. Use a friend function to add two complex no.'s and return the result as a new complex object.

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 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;
```

```
int main (){
  C1. accept ();
  C2. accept ();
Complex sum = add (C1, C2);
  cout << " The sum is ":";
  sum. display ();
    return 0;
    }
```

5. Create a class Student with private data member : name & three subjects marks. Write a friend function calculate (Average) (student) that calculates the avg. marks.

```
#include <iostream.h>
  Using namespace std;
  Class Student {
      String n;
      int m[3];
  public;
      void accept ()
        cout << " Enter Student name";
        cin >> n;
        cout << " Enter marks from
                three subjects ";
        cin >> m[0] >> m[1] >> m[2];
  } friend void calculateAvg. (Student s);
  } s;
      void calculate Average (student){
          double avg = (s.m[0] + s.m[1] + s.m[2]
                                    3.0;
  cout << "Student. " << s.n << end';
  cout << "Average Marks: " << avg << end';
  }
  int main (){
    s.accept ();
    calculate Average (s);
     return 0;
    }
```

6. Create three classes: Alpha, beta, Gamma each with private data member. Write a single friend func. that can access all three and print their sum.

Code:

```
#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,
};
Class Beta {
int n; public:
  void accept (){
  cout << "Enter value for Beta
  cin >> n;
}friend int sum (Alpha a, Beta b,
                 C);
};
int Sum (Alpha a, Beta b, Gamma
  return a.n + b.n + c.n; }
int main (){
  a.accept ();
  b.accept ();
  c.accept ();
  cout << "Sum: " << Sum (a, b, c) << endl;
  return 0;
}
```

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

Code:-

```
#include<iostream.h>
  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);
    return 0;
}
```

8. Create two classes: BankAccount and Audit. BankAccount holds private balance info. Write a friend func. in Audit that accesses & prints balance information for auditing.

Code:-

```
#include<iostream.h>
using namespace std;
class BankAccount;
class Audit {
public:
    void pBal(BankAccount);
};
class BankAccount {
double b; public:
    void accept(){
    cout<<"Enter bank acc balance";
    cin>>b;
    }friend void Audit::pBal(BankAcc);
};ba;
void Audit::pBal(BankAccount){
    cout<<"Auditing... BankAcc Balance:"
        <<ba.b<<endl;
}
int main(){
    ba.accept();
    aud.pbal(ba);
    return 0;
}
```

12/8/25

2-05-2025
Tuesday

Experiment - 5

1. Write a C++ program to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructors.

(Parametrized Constructor)

```
#include <iostream>
using namespace std;
class number {
private:
    int num;
    int sum;
public:
number(int n){
    num = n;
    sum = 0;
if (num < 0) {
Cout<< Input must be negative
    number. Setting num to 0."
    <<endl;
    num = 0;
}
for (int i = 1; i<=num; i++)
  {
    sum = sum + i;
  }
}
  void displaysum()
{
  cout << "The sum of numbers
        from 1 to " << num << " is " <<
        sum << endl;
      }
};

int main () {
  number S1(5);
    S1. displaysum();

    return 0;
}
```

(Default Constructor)

```
#include <iostream>
using namespace std;
class number
{
private:
    int num;
    int Sum;
public:
    number()
    {
        num = 0;
        Sum = 0;
    }
    void calculateand displaySum
                (int n);
{
    num = n;
    Sum = 0;

If (num < 0) {
    Cout << "Input must be a
    non-negative number. Setting
        num to 0." << endl;
    num = 0; }
```

```
for (int i = 1, i <= num; i++)
{
    Sum = Sum + i;
}
Cout << "The Sum of numbers from 1 to"
    << num << " is " << Sum << endl;
    }
};
int main () {
    number S1;
    int userNum;
    Cout << "Enter a non-negative no: ";
    Cin >> userNum;
    S1. calculate and displaySum
            (userNum);
    return 0;
}
```

2. 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.

Code : {Default Constructors}

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

  Class Student {
   private:
     String name;
     float percentage;
   public:
     Student() {
     name = "N/A";
     percentage = 0.0;
   }
    void setStudentData (String name, float spercentage)
   {  name = SName;
      Pescentage = sPercentage;
   }
```

```
void displaydata () {
cout << " Student Name: " << name << endl;
cout << " Percentage: " << percentage << "%" <<
      endl;
  }
};
  int main () {
  string studentName;
  float studentPercentage;
  S1.setstudentdata (studentName, studentPerc
  cout << "\n -- Student data -- \n");
  S1. displaydata ();
  return 0;
  }
```

(Parametrized Constructors);

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

Class Student {
  private:
     string name;
     float percentage;
  public:
  Student (string SName, float Spercentage)
  {
    Name = SName;
    Percentage = Spercentage;
  }
    void displaydata ()
    {
      cout << " Student Name: " << name << endl;
      cout << " Percentage: " << percentage << endl;
    }
};
    int main ()
      string studentName;
      float studentPercentage;
  cout << " Enter Student's Name: "
        getline (cin, studentName);
  cout << " Enter Students Percentage: " ;
      cin >> studentPercentage;
      Student S1 (studentName, studentPercentage);

  Cout << " \n -- Student data -- \n ";
      S1. displaydata ();
      return 0;
  }
```

Q- Define a class "college" member variables as roll_no: , name, course. WAP using contructors with default values as "Computer Engineering" for course. Accept this data for two objects of class and display the data.

Code: {Default Constructors}

```
#include <iostream>
#include <string>
    using namespace std;
  class college {
  public:
     int roll_no;
     String name, course;

college (int r, string n, string C = "Computer
      Engineering") {
   roll_no = r;
   name = n;
   course = C;
}
   void display () {
cout << "Roll_no" << roll_no << "Name:" << name <<
  "course: " << course << endl;
}
```

```
int main () {
College S1, (1, "HORCUS"); S2(2, "Tenz",
        "Mechanical Engineering");
S1.display(); S2.display();
return 0;
}
```

(Parametrized Constructors)

```
#include <iostream>
#include <string>
using namespace std;
class college {
public:
  int roll_no;
  String name, course;
college (int r, string C = "Computer Engineering")
{
   roll = r;
   name = n;
   course = C;
}
void display() {
cout << "Roll_no:" << roll_no << "Name:" <<
  name << "Course" << course << endl;
}
};
```

Experiment - 6

Q1. Create a base class called Person with attributes name and age. Derive a class Student from Person that adds an attribute Roll-no. Write a function to display all details of the student.

{SINGLE INHERITANCE}

#Code

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

class Person {
protected:
string name;
int age;
public:
Person(const string& n, int a): name(n), age(a){}

string getName() const {
return Name;
}
int getAge() const {
return Age;
}
};
class Student : Public Person {
private:
int roll_number;

public:
Student (const string& n, int a,
int r): Person(n,r),
roll_number(r){}
void displaydetails() const{

cout << "Student Details:" <<endl;
cout << "Name:" << getName ()<<endl;
cout << " Age:" << getAge ()<<endl;
cout << "Roll_nomber:"<<endl;
}
};
int main (){
Student.S1("Max",19,16);
S1. display details()
return 0;
}
```

Q2- Multiple Inheritance :-
Create two base class Academic and Sports.
Academic class contains marks of a student.
Sports class contains Sports score.
Create a derived class result that inherits from both Academic and Sports.
Write a function to calculate the total score and display details.

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

class Academic {
  Protected:
    int marks;
  public:
   Academic (int m) : marks (m) { }
   int getmarks () const { return marks;}
};

class Sports {
   Protected:
     int sportsScore;
   Public:
   Sports (int s) : SportsScore (s) { }
    int getSportsScore () const { return
          SportsScore;
    };
class Result : Public Academic, public
                  Sports {
   private:
     String name;

Public:
   Result (const string &n, int m, int s)
        : Academic(m), Sports (s), name(n)
     { }

   int calculateTotalScore () const {
      return Marks + SportsScore;
   }
      void displaydetails () const {
Cout << " Student Details" << endl;
Cout << "Name :" << name << endl;
Cout << "Academic Marks:" << getMarks () << endl
Cout << "Sports Score:" << getSportsScore () << endl;
Cout << "Total Score:" << calculateTotalScore
                          << endl;
};
```

```
int main() {
    StudentResult result("Max Doo", 85, 90);
    result.displayDetails();

    return 0;
}
```

Q3- Multilevel Inheritance:
Create a class Vehicle with attributes like brand & model. Derive a class ~~model~~ Car from vehicle which adds an attribute type (eg: sedan, SUV, truck). Further derive a class ElectricCars from car which adds battery capacity. Write functions to display all the details.

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

class Vehicle {
protected:
    string model;
    string brand;
public:
    Vehicle(const string& b, const string& m)
        : brand(b), model(m) {}

    void displayVehicleDetails() const {
        cout << "Brand: " << brand << endl;
        cout << "Model: " << model << endl;
    }
};

class Car : public Vehicle {
protected:
    string Type;
public:
    Car(const string& b, const string& m,
        const string& t)

    void displaycardetails() const {
        displayVehicle details();
        cout << "Type:" << type << endl;
    }
};

class ElectricCar : public Car {
private:
    int batteryCapacity; // in KWh
public:
```

```
ElectricCar(const string& b, const string& m,
            const string& t, int cap):
    Car(b, m, t), batteryCapacity(cap) {}

void displayAllDetails() const {
    cout << "Electric Car details:" << endl;
    displayCarDetail();
    cout << "Battery Capacity: " << batteryCapacity
    << "kWh" << endl;
}
};

int main() {
    ElectricCar tesla("Tesla", "Model S",
                      "Sedan", 100);
    tesla.displayAllDetails();
    return 0;
}
```

Q4- Hierarchical Inheritance :-

Create a base class Employee with attributes emp_Id and Name. Derive two classes Managers and developer from employee. Manager has an attribute department and developer has an attributes programming langu. Write functions to display details for both.

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

class Employee {
protected:
    int EmpID;
    string name;
public:
    Employee(int id, const string& n):
        empID(id), name(n) {}
    void displaydetailEmployee() const {
    cout << "Employee Id" << empID << endl;
    cout << "Name:" << name << endl;
}
};

class Manager : public Employee {
private:
    string department;
public:
    Manager(int id, const string& n, const
        string& dept) : Employee(id, n)
        department(dept) {}
    void displayEmpdetails() const {
cout << "Managers details" << endl;
```

```
displayEmployeeDetails();
Cout << "Department" << department << endl;
}
};
Class Developer : public Employee {
private:
  String programmingLang;
public:
  Developer(int id, const string& n,
            const string& lang) :
  Employee(id, n) programmingLang(lang){}
void displaydetails() const {
Cout << "Developer Details" << endl;
  displayEmpdetails();
Cout << "Programming lang: " << programming
  lang << endl;
}
};
  int main(){
Manager M1(1245, "Anonymous", "Marketing");
Developer D1(1298, "Anonymous2", "C++");
  M1.displaydetails();
  Cout << endl;
  D1.displaydetails();
    return 0;
  }
```

Experiment – 07
(Polymorphism)

Q. Write a program to demonstrate the compile time polymorphism (Function Overloading and Operator Overloading – Unary).

(i) Write a program using function Overloading to calculate the area of a laboratory (which is rectangular in shape) and area of classroom (which is square in shape).

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

int calculateArea(int side){
    return side * side;
}
    int calculateArea(int length, int
        width){
        return length * width;
}

Class Counter {
    private
    int count;
public:
    Counter() : count(0) {}

Counter operator++(){
    ++count;
    return *this;
}
    void display(){
    Cout << "Current count is: "<< count<<endl;
}
};

int main(){}

void display(){
Cout <<"Current count is:"<<count <<endl;
}
};

int main(){
Cout <<" Area Square is: "<<calculateA
                (10) <<endl;
Cout <<" Area Rectangle:" <<calcula
                (15,8)<<endl;
Counter C;
C.display();
(++(++C)).display();
return 0;
```

(ii) Write a program using function overloading to calculate the sum of 5 float values and sum of 10 integer value.

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

float sum_values(float arr[], int size){
    if(size != 5){
    cout << "Warning: This function is
    intended for 5 float values." << endl;
    }
    float total = 0.0f;
    for(int i = 0; i++; i<size){
        total += arr[i];
    }
    return total;
    }

int Sumvalues(int arr[], int size){
    if(size != 10){
    cout << "Warning: This function is
    indeuted for 10 integers values" <<
        endl;
    }
    int total = 0;
    for(int i=0; i<size; i++){
        total += arr[i];
    }
    return total;
    }
    int main(){
    const int F=5, I=10;
    float f_val[F];
    int i_val[I];

cout << "Enter integer value:" << F << "Float
    \n";

for(i=0; i<F; ++i){
    cout << "#" << i+1 << ":", cin >> fval[i];
}

cout << "\nEnter " << "I" << " Ints:\n";
    for(int i=0; i<I; ++i){
        cout << "#" << i+1 << ":", cin >> i
    }
cout << "\nResults:\n"
<< "Float sum(" << f << "): " << sum_v
        (f_val, f) << "\n";
<< "Int sum(" << I << "): " << sum_val
        (i_val, I) << endl;
    return 0;
```

(iii) Write a program to implement unary (-) operator when used with the object so that the numeric data members of the class is negated.

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

class Number {
    private:
        int value;
    public:
        Number(int val) : value(val) {}
        Number operator-() const {
            return Number(-value);
        }
        void display() const {
            cout << "value: " << value << endl;
        }
};

int main() {
    Number my_number(15);
    cout << "Original object's value" << endl;
    my_no.display();
    No.negated = -my_no;
    cout << "\nAfter Applying the unary - operator."
         << endl;
    negated_no.display();

    cout << "\n Original object value is still" <<
            endl;
    my_no.display();

    return 0;
}
```

Experiment — 8

a. Operator Overloading :

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

Output :-

```
firstString:
Enter String: abc
Secoud String:
Enter String: xyz.
Concatenated String: abc xyz.
```

b. WAP to create a base class. I login having data members names password. Declare accept() function virtual. Derive email login and membership login class from I login. Display email login details and membership login details of the employee.

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

class I login {
    protected:
        String name, password;
    public:
        virtual void accept () {
            Cout << "Enter name:";
            Cin >> name;
            Cout << "Enter password:";
            Cin >> password;
        }
virtual void display () {
    Cout << "Name: " << Name << endl;
    Cout << "Password: " << password << endl;
}
```

```
class Emaillogin : public I login {
    String email;
public:
    Void accept () override {
        Cout << "Enter email ID:";
        Cin >> email;
        I login :: accept ();
    }
    void display() override {
        Cout << "In --- Email logindetails ---"
             << endl;
        Cout << "Email ID: " << email << endl;
        I login :: display ();
    }
};

class Membership login : public I login {
    String member ID;
public:
    Void accept () override {
        Cout << "Enter membership ID:";
        Cin >> member ID;
        I login :: accept ();
    }
    void display () override {
        Cout << "In --- Membership login
             << endl;
```

```
cout << "Membership ID" << memberId << endl;
    I.login::display();
}
};
int main(){
    I.login * login;

    Email login e;
    membership login m;

    login = &e;
    login -> accept();
    login -> display();

    login = &m;
    login -> accept();
    login -> display();

    return 0;
}
```

4111

Q1 → Write a program to copy the contents one file into another. Open 'First.txt" in read mode and "Second.text". Assume "First.txt is already created.

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

int main(){
    if stream in File("First.txt");
    of stream out File("Second.txt");

    if(!infile){
    cout << "Error opening First-txt" << endl;
        return 1;
    }
    char ch;
    while(infile.get(ch)){
        outfile.put(ch);
    }
    cout << "File copied succenfully" <<
    infile.close();
    outfile.close();
    return 0;
}
```

Q2⇒ WAP to Count Digits and Spaces using file handling.

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

int main () {
  ifstream file ("First.txt");
  if (!file) {
    cout << "Error opening file" << endl;
    return 1;
  }
    char ch;
    int digits = 0; spaces = 0;

  while (file.get(ch)) {
    if (isdigit(ch))
      digits++;
    else if (isspace(ch))
      spaces++;
    }
  Cout << "Digits: " << digits << endl;
  Cout << "Spaces: " << spaces << endl;
  file.close();
    return 0;
  }
```

Q3⇒ WAP to Count words using file handling.

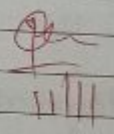
Code ⇒

```
#include <iostream>
#include <fstream>      // Reading from and writing to
#include <string>

using namespace std;

int main () {
  ifstream inputfile ("input.txt");

if (! inputFile.is_open()) {
    cerr << "Error: Could not open the
            file input.txt" << endl;
    return 1;
  }

int wordCount = 0;
 String word;

while (inputfile >> word) {
      wordcount++;
    }
      inputfile.close();
Cout << "Total no of words in the file:
      wordcount << endl;
return 0; }
```

d) 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() {
    string searchword;
    cout << "Search for : ";
    cin >> searchword;

    ifstream file("input.txt");

    if (!file.is_open()) {
        cerr << "Error : File not found." <<
                endl;
        return 1;
    }
    int count = 0;
    string currentword;
    while (file >> currentword) {
        if (currentword == searchwor
    { count++; }
```

```
    file.close();

    cout << "' " << searchword << " ' occured " <<
    cout << " Times. " << endl;
    return 0;
}
```



Experiment – 10

Q – Function Template

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

template <typename T>
T sumArray (const T arr[], int size) {
    T sum = 0;
    for (int i = 0; i < size; ++i) sum +=
        arr[i];
    return sum;
}
int main () {
const int S=10;

int iA[S]; float fA[S]; double dA[S];

Cout<<" Please enter 10 integer values:\n";
for( int i = 0 ; i<S ; i++) {
    Cout<<" Int [ "<<i+1<<" / "<<S<<" ]:
    cin >>iA[i];
}
Cout <<" Please enter 10 floating values:\n";
for (int i = 0; i<S; i++) {
    Cout <<" float [ "<<i+1<<" / "<<S<< " ]:";
    cin >> fA[i];
}

Cout <<" \n Please enter 10 double value
    :\n" );
for (int i = 0; i<S; i++) {
    Cout <<" Double [ " <<i+1<<" / " <<
        S<< " ]:
    cin >> dA[i];

cout << "\n--- Calculated Values ---:
Cout << " IntSum: <<SumArray (iA,S) << "\n";
Cout << " floatSum: << fixed; sumArray(fA,S);
Cout << " "DoubleSum"; << fixed; sumArray(dA,S);

    return 0;
}
```

Q- Square function using template specialization:-

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

template <typename T>
T square(T val){
}
template <>
string square <string> (string srr){
    return srr + srr;
}
int main(){
    int intval = 12;
    string strval = "Short";
    float floatval = 2.5f;

cout << "Int Square" << square(intval) << "\n";
cout << "String Square" << square(strval) <<
cout << "float Square" << square(floatval) <<
    return 0;
}
```

Comp-11

Q- WAP to create a vector, modify the value multiply by a scalar value, display the vector.

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

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

Exp - 11
(With Iterators)

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

int main(){

vector<int> v = {1,2,3,4,5,6,7,8,9,
                 10};
cout<<"Initial vector:"<<endl;
for(vector<int>::iterator it=v.begin();
   it!=v.end();++it){
      cout<<*it<<" ";
   }

cout<<endl;
cout<<"Multiply by 10"<<endl;
for(vector<int>::iterator it=v.
      begin(); it!=v.end();
                ++it){
   *it=(*it)*10;
}
cout<<endl;
   return 0;
}
```

Exp - 12

A) Implement Stack

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
  stack<char> cars;
   cars.push("BMW");
   cars.push("Audi");
   cars.push("Mercedes");
   cars.push("Ferrari");
cout<<"Top element is:"<<cars.top()<<endl;
cout<<size of stack is :<<cars.size()<<
                                  endl;

   cars.pop();
   cars.pop();
{while(!cars.empty())
   cout<<"Elements in stack are:"<<
            cars.top()<<endl;
      {cars.pop();
}return 0;
```

O/P :- Top element : ferrari
Size of Stack : 4
Elements in Stack are : Audi
" — " : BMW.

b) Implement Queue :-

```
#include <bits/stdc++.h>
using namespace std;
int main(){
queue<int> age;
age.push(21);
age.push(22);
age.push(23);
age.push(24);

cout<<"Back element is : "<<age.back()<<endl;
cout<<"front elements is : "<<age.front()<<endl;

age.pop();
age.pop();
while(!age.empty())
{
cout<<"elements in queue are:"<<age.front()<<endl;
age.pop();}
return 0;}
```

O/P

front element = 21
Back element = 24

elements in queue : 23
elements in queue : 24