Practical - 1

Q1 Write a C program for the addition of 2 numbers

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

int main(){
    int a, b;

    cout << "Enter the first number" << endl;
    cin >> a;

    cout << "Enter the second number"
                                << endl;
    cin >> b;

    cout << "The sum of the 2 numbers
                is = " << a+b << endl;
}
```

Q2 Write a C++ program to check if a number is even or odd.

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

int main (){
    int num;
    cout << "Enter the first number" << endl;
    cin >> num;

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

Q3 Write a C++ program to perform basic arithmetic operations

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

int main () {
    int num1, num2;
    char operation;

    cout << Enter number : " << endl;
    cin >> num1;

    cout << Enter number 2 : " << endl;
    cin >> num 2;

    cout << " Enter the operation
            (+, -, *, /) " << endl;
    cin >> operation;

    switch (operation) {
    case '+';
    cout << " Sum = " << num1+num2<<endl;
    break;

    case '-' :
    cout << "Subtraction = "<< num1 - num2;
    break;
```

```
case '*':
cout << " ~~Subtraction~~ Product = " << num1 * num2
                                << endl;
break;

case '/':
cout << " Remainder = " << num1 % num2
                                << endl;
break;
}
}
```

Q4 Write a C++ program to print 0-10 numbers

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

int main () {
    for (int i; i <=10; i++) {
        cout << i << endl;
    }

}
```

Q5 Write a C++ program to print 10-0 numbers

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

int main () {
    for (int i = 10; i >= 0; i--) {
        cout << i << endl;
    }
}
```

Q6 Write C++ programs to print the following patterns.

a)
```
    *
  * *
* * *
```

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

int main() {
    int rows = 5;

    for(int i = 1; i <= rows; i++){
        for(int j = 1; j <= rows - i; j++){
            cout << " ";
        }
        for(int k = 1; k <= i; k++){
            cout << "*";
            if (k < i) cout << " ";
        }
        cout << endl;
    }

    return 0;
}
```

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

```
#inclu
using

int ma
    in
    co
    ci
```

c)
```
1
1 2
1 2
1 2
```

b) 1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

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

int main() {
    int num;
    cout << "Enter first number" << endl;
    cin >> num;

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

c) 1
1 2 3
1 2 3
1 2 3 4

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

int main () {
    int num;
    cout << "Enter the first number" << endl;
    cin >> num;

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

8/7/25

15/7/25
Tuesday

Experimen

Q1> WAP to
having
Accept &

```
#include
using nam

class Stu
private:
    strin
    int

public:
    void
        co
        ge
        c
        ci
    }
    void
        c
        c
        c
    }
};
```

19/7/25
Tuesday

Experiment - 1

Q1> WAP to declare a class 'Student' having data members as name, Rollno Accept & display data for one student

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

class Student {
private:
    string name;
    int rollNumber;

public:
    void acceptData() {
        cout << "Enter student name: ";
        getline(cin, name);
        cout << "Enter roll number: ";
        cin >> rollNumber;
    }
    void displayData() {
        cout << "In StudentDetails: " << endl;
        cout << "Name: " << name << endl;
        cout << "Roll Number: " << rollNumber << endl;
    }
};
```

```
int main(){
    student s;

    s.acceptData();
    s.displayData();

    return 0;
}
```

Output:
Enter student name
: Anushree
Enter roll number:
36
--- Student Details ---
Name : Anushree
Roll no : 36

Q2 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{
    private:
        int id;
        string name;
        float price;

Public:
    void acceptData(){
        cout << "Enter BookID: ";
        cin >> id;
        cin.ignore();
        cout << "Enter Book Name: ";
```

```
    getline(cin, name);
    cout << "Enter Book Price: ";
    cin >> price;

void displayData(){
    cout << "\n--- Book Details -- " << endl;
    cout << " ID: " << id << endl;
    cout << " Name: " << name << endl;
    cout << " Price: ₹ " << price << endl;

  float getPrice(){
      return price;
  }
};

int main(){
    Book b1, b2;

    cout << "\nEnter details for Book1:\n";
    b1.acceptData();

    cout << "\n Enter details for Book2:\n";
    b2.acceptData();

    cout << "\n Book with greater price is:\n";

    if(b1.getPrice() > b2.getPrice()){
        b1.displayData();
    }else{
```

```
    b2.displayData();
  }

    return 0;
}
```

Output:
Enter details of Book 1:
Enter Book ID : 12363874
Enter Book Name: dfhb
Enter Book Price: 399

Enter details for Book 2:
Enter Book ID : 128473287
Enter Book Name: dffjgef
Enter Book Price: 249

Book with greater price is:
--- Book details ---
ID: 12363874
Name: dfhb
Price: 399

Q3 WAP to declare a class time having data members as H, M & S. Accept data for one object & display total time in seconds.

```
#include <iostream>
using namespace std;
class Time {
private:
   int H, M, S;
Public:
   void accept Data(){
      cout << "Enter Hours: ";
      cin >> H;
      cout << "Enter Minutes: ";
      cin >> M;
      cout << "Enter Seconds: ";
      cin >> S;
   }
   int totalSeconds(){
      return (H*3600)+(M*60)+S;
   }
   void displayTotalSeconds(){
      cout << "\n Total Time in seconds:"
                << totalSeconds() << "sec"
                << endl;

   };
```

```
int main (){
    Time t1;
    t1.acceptData ();
    t1.displayTotalSeconds();
    return 0;
}
```

Output:
Enter Hours : 1
Enter Minutes : 1
Enter Seconds : 1

Total time is seconds : 3661 sec

15/7

Experiment - 2

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

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

class City {
public:
    string name;
    int population;

    void input(){
        cout << "Enter city name: ";
        cin >> name;
        cout << "Enter population: ";
        cin >> population;
    }
};
int main(){
    City c[5];
    int maxIndex = 0;

    for (int i=0; i<5; i++){
        cout << "\n City " << i+1 << ": \n";
        c[i].input();
        if (c[i].population > c[maxIndex].
                                population){
```

```
        maxIndex = i;
    }
}

cout << "\nCity with highest population:"
                <<c[maxIndex].name
                <<endl;

return 0;
}
```

Q 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 & display them.

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

class Account{
public:
    int accNo;
    float balance;

    void input(){
        cout << "Enter account number: ";
        cin >> accNo;
        cout << "Enter balance : ";
        cin >> balance;
    }

    void giveInterest(){
        if (balance >= 5000){
            balance += balance * 0.10;
        }
    }

    void display(){
        cout << "Account No: " << accNo << " | Balance: "
        << balance << endl;
```

```
    }
};

int main(){
    Account a[10];

    for(int i=0; i<10; i++){
        cout<< "\nAccount "<< i+1<< ": \n";
        a[i].input();
        a[i].giveInterest();
    }

    cout<< "\nAccounts with balance>=5000
                    after interest: \n";
    for(int i=0; i<10; i++){
        if (a[i].balance >=5000)
            a[i].display();
    }

    return 0;
}
```

~~Output~~

Q3 WAP to declare a class 'Staff' having data members as name & post. Accept this data for 5 staff and display names of staff who are "HOD".

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

class Staff{
    string name;
    string post;

Public:
    void input(){
        cout << "Enter Name: ";
        cin << "Enter Post: ";
        cin >> post;
    }

    void display If HOD(){
        if (post=="HOD"||Post=="hod"){
            cout << "HOD Name " << name << endl;
        }
    }
};

int main()
    Staff s[5];
```

```
for (int i=0; i<5; i++){
    cout << "\n Enter details of staff " <<
    s[i].input();          i+1 << endl;
}

cout << "\n list of HODs : \n";
for (int i =0; i<5 ; i++){
    s[i].display IfHOD();
}

return 0;
}
```

29/7/25

Experiment - 3

Q1 Write a program to declare a class 'book' containing data members as book_title, author_name & price. Accept and display the information for one object using a pointer to that object.

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

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

public :
    void accept () {
        cout << "Enter book title: ";
        getline (cin, title);
        cout << "Enter author name: ";
        getline (cin, author);

        cout << "Enter price:";
        cin >> price;
        cin.ignore();
    }
```

```
    void display(){
        cout << "\n Book Details: \n";
        cout << "Title : " << title << endl;
        cout << "Author: " << author << endl;
        cout << "Price: ₹" << Bprice << endl;
    }
};

int main(){
    Book *ptr;
    Book b;
    ptr = &b;

    cout << "Enter details for the book:\n";
    ptr -> accept();

    cout << "\n Displaying book details:\n";
    ptr -> display();

    return 0;
}
```

Q WAP to declare a class 'student' having data members roll_no & percentage. Using 'this' pointer invoke member function to accept and display this data for one object of the class

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

class Student {
private:
    int roll_No;
    float percentage;

public:
    void accept(){
        cout << "Enter Roll Number: ";
        cin >> this -> roll_no;

        cout << "Enter Percentage: ";
        cin >> this -> percentage;

    void display(){
        cout << "\nStudent Details:\n";
        cout << "Roll Number: " << this -> roll_
                                no << endl;
        cout << "Percentage: " this -> percentag
                                << "%." << endl;
    }
};
```

```
int main (){
    student s;

    s.accept ();
    s. display ();

    return0;
}
```

Q
3/9

Experim

Q Create
each w
a friend
private
return

inclu
using

class B

Class A
private
in

public :
A

}

}

};

class B
private
int
public
B

Experiment - 4

Q Create 2 classes, Class A and 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;

Class A {
private:
    int x;

public:
    A(int val){
        x = val;
    }

    friend int sum(A,B);
};

class B {
private:
    int y;
public:
    B(int val){
        y = val;
    }
```

```
    friend int sum (A, B);
};

int sum (A a, B b){
    return a.x + b.y;
}

int main()
    A obj1 (5);
    B obj2 (7);

    cout << "Sum: " << sum(obj1, obj2) << endl;

    return 0;
}
```

→ Output:
Sum: 12

Q2 WAP witha class Number that contains aprivate integer. Use a friend function swapNumbers(Number&, Number&) to swap the private values of two Number objects

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

class Number{
private:
    int value;
public:
    Number(int val){
        value = val;
    }

    void show() {
        cout << "Value: " << value << endl;
    }

    friend void swapNumbers
                (Number&, Number&);
};

void swapNumbers(Number&n1, Number& n2){
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp;
}
```

```
int main(){
    Number a(10);
    Number b(20);

    cout << "Before swap: " << endl;
    a.show();
    b.show();

    swapNumbers(a, b);

    cout << "After swap: " << endl;
    a.show();
    b.show();

    return 0;
}
```

->Output:

```
Before swap:
Value: 10
Value: 20
After swap:
Value: 20
Value: 10
```

Q3 Define 2 classes Box and Cube, each having a private volume. Write a friend function findGreater(Box, Cube) the determines which object has a larger volume.

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

class Cube;

class Box{
private:
    int volume;
public:
    Box(int v){
        volume = v;
    }
    friend void findGreater(Box, Cube);
};

class Cube{
private:
    int volume;

public:
    Cube(int v){
        volume= v;
    }
```

```
    friend void find Greater (Box, Cube);
};

void find Greater (Box b, Cube c){
    if (b.volume > c.volume){
        cout << "Box has greater volume: " <<
    } else if (c.volume > b.volume){ b.volume<<
        cout << "Cube has greater volume: " << c.volume
                                                <<endl;
    } else {
        cout << "Both have equal volume: " << b.volume
                                                <<endl;
    }
}

int main(){
    Box box1 (100);
    Cubebox2 (90);

    find greater (box1, cube1);

    return 0;
}
```

Output:
Box has greater volume: 100

Q4 Create a class Complex with real and 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:
    int real;
    int imag;

public:
    Complex (int r = 0; int i = 0){
        real = r;
        imag = i;
    }

    void show (){
        cout << real << " + " << imag <<
                                " i " << endl;
    }

    friend Complex add (Complex, Complex);
};
```

```
Complex add (Complex c1, Complex c2){
    Complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;
    return result;
}

int main (){
    Complex a(2, 3);
    Complex b(4, 5);

    Complex c = add(a, b)

    cout<< "Sum of complex no.: ";
    c.show();

    return0;
}
```

→ Output:
Sum of complex numbers : 6+ 8i

Q5 Create a class Student with private data members : name and 3 subject marks. Write a friend function calculateAverage (Student) that calculate and displays the average marks.

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

class Student {
private:
    string name;
    int mark1, mark2, mark3;

public:
    Student (string n, int m1, int m2, int m3){
        name = n;
        mark1 = m1;
        mark2 = m2;
        mark3 = m3;
    }
    friend void calculateAverage (Student);
};

void calculateAverage (Student s){
    float avg = (s.mark1 + s.mark2 + s.mark3)/3;
    cout << "Student name: " << s.name << endl;
    cout << "Average marks: " << avg << endl;
}
```

```
int main (){
    Student s1 ("Ani", 85, 90, 95);
    calculateAverage (s1);

    return0;
}
```

→ Output:
Student Name : Anushre
~~Student~~ Average Marks : 90

0, 95);

Q6 Create three classes: Alpha, Beta, and Gamma, each with a private data member. Write a single friend function that can access all three and print their sum

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

class Beta;
class Gamma;

class Alpha{
private:
    int a;

public:
    Alpha(int val){
        a = val;
    }

    friend void totalSum(Alpha,Beta,Gamma);
};

class Beta{
private:
    int b;

public:
```

```
    Beta(int val){
        b = val;
    }

    friend void totalSum(Alpha, Beta, Gamma);
};

class Gamma{
private:
    int c;

public:
    Gamma(int val){
        c = val;
    }

    friend void totalSum(Alpha, Beta, Gamma);
};

void totalSum(Alpha x, Beta y, Gamma z){
    int sum = x.a + y.b + z.c;
    cout << "Sum of all values : " << sum << endl;
}

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

```
    totalSum(obj1, obj2, obj3);

    return 0;
}
```

-> Output:
Sum of all values : 60

Q7 Create a class Point with private members x and y. Write a friend function that calculates and returns the distance between 2 Point objects

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

class Point {
private :
    int x, y;

public :
    Point (int xVal, int yVal){
        x = xVal;
        y = yVal;
    }

    friend double calculateDistance(Point, Point);
};
```

```
double calculateDistance (Point p1, Point &p2)
    int dx = p2.x - p1.x;
    int dy = p2.y - p1.y;
    return sqrt (dx*dx + dy*dy);
}

int main(){
    Point p1 (3, 4);
    Point p2 (7, 1);

    double distance = calculateDistance (p1, p2);

    cout << "Distance between points: " <<
                                distance << endl;

    return 0;
}
```

→ Output:
Distance between points.

Q8 Create two classes : BankAccount and 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 Audit;

class Bank account{
private:
    int balance;

public:
    BankAccount (int b){
        balance = b
    }
    friend void auditBalance (BankAccount, Audit);
};

class Audit {
public:
    void startAudit (BankAccount acc){
        auditBalance (acc, *this);
    }
```

```
    friend void auditBalance(BankAccount, Audit);
};

void auditBalance(BankAccount acc, Audit){
    cout << "Balance: Rs." << acc.balance << endl;
}
int main() {
    BankAccount myAcc(50000);
    Audit auditor;

    auditor.startAudit(myAcc);

    return 0;
}
```

→ Output:
Balance : Rs. 50000

Bank Account, Audit)

nt acc, Audit){
acc.balance<<endl;

0);

y Acc);

Q9 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 num;
public:
    Number (int n){num = n;}
    void swap (Number &n){
        int t = num;
        num = n.num;
        n.num = t;
    }
    void show (){cout << num << " ";}
};

int main (){
    Number a(10), b(20);
    a.show(); b.show(); cout << endl;
    a.swap(b);
    a.show(); b.show();
```

→ Output:

10 20

20 10

Q10 WAP to swap 2 numbers from same class using ~~obj~~ ~~ot~~ ~~a~~ concept of friend function.

→ Output:
10 20
20 10

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

class Number {
    int num;
public:
    Number(int n){ num=n; }

    friend void swapNumbers (Number&x,
                             Number&y);

    void show (){cout<< num << " ";}
};

void swapNumbers (Number &x, Number&y){
    int temp = x.num;
    x.num = y.num;
    y.num = temp;
}

int main(){
    Number a(10), b(20);
    a.show(); b.show(); cout<<endl;
    swapNumbers (a,b);
    a.show(); b.show();
}
```

Q11 WAP to swap 2 numbers from different class using friend function

```
#inclue iostream
using name space std;

class B;
class A {
    int numA;
public:
    A(int n){ numA = n;
    friend void swapNumbers (A&x,B&y);
    void show () { cout << numA << " "; }
};

Class B{
    int numB;
public;
    B (int n) {numB = n ;}
    friend void swapNamber (A&x, B&y);
    void show () { cout << numB << " "; }
};

void swapNumbers (A&x, B&y){
    int temp = x. numA
    x.numA = y. numB
    y.numB = temp:
}
```

```
int main(){
    A obj1(10);
    B obj2(20);

    obj1.show(); obj2.show(); cout<<endl;
    swapNumbers(obj1, obj2);
    obj1.show(); obj2.show();
}
```

→ Output:

10 20

20 10

Q12 WAP to create 2 classes Result1 and Result2 which stores the marks of the student. Read the value of marks for both the class objects and compute the average of the 2 marks

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

class Result 2;

class Result 1 {
    float marks;
public :
    void read(){
    cout << "Enter marks for Result 1: ");
    cin >> marks;
    }
    friend void average (Result1, Result2);
};

class Result2 {
    float marks;
public;
    void read() {
        cout << Enter mark for Result 2 : ");
        cin >> marks;
    }
}  friend void average(Result1, Result2);
};
```

```
void average (Result1 r1, Result2 r2){
    float avg = (r1.marks + r2.marks)/2
    cout << "Average marks = " << avg << endl;
}

int main(){
    Result1 obj1;
    Result2 obj2;

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

    average (obj1, obj2);
    return 0;
}
```

→Output:

```
Enter marks for Result1: 100
Enter marks for Result2: 99
Average marks = 99.5
```

Q5 WAP to find the greatest number among 2 numbers from 2 different classes using friend function.

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

class B;

class A {
    int numA;
public:
    void read() {
        cout << "Enter number for class A: ";
        cin >> numA;
    }
    friend void findGreatest (A, B)
};

class B {
    int numB;
public:
    void read() {
        cout << "Enter number for ClassB: ";
        cin >> numB;
    }
    friend void findGreatest (A, B);
};
```

```
void findGreatest (A x, B y){
    if (x.numA > y.numB)
        cout<< "Greatest: "<< x.numA<< endl;
    elseif (y.numB > x.numA)
        cout<< "Greatest: "<< y.numB<< endl;
    else
        cout<< "Equal" << endl;
}

int main(){
    A obj1;
    B obj2;

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

    findGreatest (obj1, obj2);
    return0;
}
```

→ Output:
Enter number for class A: 17
Enter number for class B: 23
Greatest: 23

Experiment - 5

Q1 WAP to find the sum of numbers between 1 to n using a constructor where the value

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

class number{
    int num;
public:
    number(){
        cout << " Enter a number: " << endl;
        cin >> num;

        int sum=0;
        for(int i=1; i<=num; i++){
            sum = sum+i;
        }

        cout << "Sum of number upto << num << "is
            : " << sum << endl;
    }
};

int main(){
    number n;

    return 0;
}
```

-> Output:
Enter a number: 3
The sum of numbers upto 3 is 6.

Q2 WAP to declare a class "student" having data members as name and percentage. Write a constructor to initialize these data members. Accept and display data for one student.

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

class Student{
    string name;
    float per;
    public:
    Student(string n, float p){
        name = n;
        per = p;
    }

    void display(){
        cout<< "Name: " << name << endl;
        cout<< "Percentage: " << per << endl;
    }
};

int main(){
    Student s("Anushree", 93.2);
    s.display();

    return 0;
}
```

→ Output:
Name: Anus
Percentage:

Q3 Define a clas
as roll_no
constructor
"Computer
this data fo
the data.

```
#include <io
using name sp

class Colleg
    int roll;
    string na
    string c
    public:
    College(i
        re
        na
        co
    }

    void di
        cout
        cou
    }
```

Page No.:
Date:

-> Output:
Name: Anushree
Percentage: 93.2

Q3 Define a class 'College' members variables as roll_no, name, course, WAP using constructor with default value as "Computer Engineering" for course, Accept this data for 2 objects of class and display the data.

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

class College{
    int roll;
    string name;
    string course;
    public:
    College(int r=00, string n="Unknown", string
            c="Computer Engineering"){
        roll=r;
        name=n;
        course=c;
    }

    void display(){
        cout<< "Roll Number: "<< roll<< endl;
        cout<< "Name: "<< name<< endl;
        cout<< "Course: "<< course<< endl;
```

```
};

int main(){
    College s1(37, "Anushree");
    College s2(18, "Gaurav");

    s1.display();
    s2.display();

    return 0;
}
```

→ Output:
Roll number: 37
Name: Anushree
Course: Computer Engineering
Roll number: 18
Name: Gaurav
Course: Computer Engineering

Q4 WAP to demostrate constructor overloading

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

class Student{
    int roll;
    string name;
    public:
```

```
Studen
    no
    re
}

Studen
    no
    re
}

Studen
    n

void
    c
    co
}
};

int main(){
    Student
    Student s
    Student s

    s1.displa
    s2.displa
    s3.displa
    return 0;
}
```

```
    Student(){
        name = "Unknown";
        roll = 0;
    }

    Student(string n){
        name = n;
        roll = 0;
    }

    Student (string n, int r){
        name = n;
        roll = r;

    void display (){
        cout<< "Name: " << name << endl;
        cout<< "Roll Number: " << roll << endl;
    }
};

int main(){
    Student s1;
    Student s2("Anushre");
    Student s3("Gaurav", 18);

    s1.display();
    s2.display();
    s3.display();
    return 0;
}
```

→ Output:
Name : Unknown
Roll number: 0
Name : Anushre
Roll number: 0
Name : Gaurav
Roll number: 18

Experimen

Q1 Write a p
inheritan

#include
using name

class dep
protected:
strin
};

class stu
protected
string
int ro
};

class mark
int
public :
void

Experiment - 6

Q1 Write a program to implement multilevel inheritance. Assume suitable data.

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

class department{
protected:
        string dname;
};

class student: protected department{
protected:
    string sname;
    int roll;
};

class marks: protected student{
        int m1, m2, percentage;
public:
        void accept(){
            cout<< "Enter department:" << endl;
            cin >> dname;
            cout << "Enter name:" <<endl;
            cin >> sname;
            cout << "Enter marks1;" << endl;
            cin >> m1;
            cout << Enter marks2:" << endl;
            cin >> m2;
```

```
void calculate(){
    int per = (m1+m2)/2
    cout<<"Department: "<<dname<<endl;
    cout<<"Name: "<<sname<<endl;
    cout<<"Percentage: "<<per<<endl;
  }
};

int main(){
    marks m;
    m.accept();
    m.calculate();

    return0;
}
```

→Output:

```
Enter department:
Anushree
Enter marks1:
88
Enter marks2:
91
Department : CSE
Name: Anushree
Percentage: 89
```

Q2 WAP to implement multiple inheritance. Assume suitable data.

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

class department{
protected:
        string dname;
};

class student{
protected:
        string sname;
        int roll;
};

class marks: protected department, protected
                                    student{
        int m1, m2, percentage;
public:
        void accept(){
            cout<<"Enter department:"<<endl;
            cin >> dname;
            cout<<"Enter name: "<<endl;
            cin >> sname;
            cout<<"Enter marks1:"<<endl;
            cin>> m1;
            cout<<"Enter marks2:"<<endl;
            cin>> m2;
```

```
}
        void calculate(){
            int per = (m1+m2)/2;
            cout << "Department: " << dname;
            cout << "Name: " << sname << endl;
            cout << " Percentage: " << per;
        }
};

int main(){
    marks m;
    m.accept();
    m.calculate();
}
```

Q3 WAP to implement hierarchical inheritance

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

class Person{
    protected:
    string name;
    int age;
    public:
    void acceptPer(){
        cout << "Enter name: " << endl;
        cin >> name;
        cout << "Enter age: " << endl;
        cin >> age.
}
```

```
-m2) /2;
ment: "<< dname<<
<< sname<< endl;
age : "<< per<<endl;

ical inheritance

"<< endl;

"<< endl;
```

```
};

class Student : public Person{
    int roll;
    float per;
    public:
    void acept Stud(){
        cout << " Enter roll number:" << endl;
        cin >> roll;
        cout << " Enter Percentage:" << endl;
        cin >> per;
    }

    void display Stud(){
        cout << "Name: " << name << endl;
        cout << "Age : " << age << endl;
        cout << "Roll No. : " << roll << endl;
        cout << "Percentage : " << per << endl;
    }
};

class staff : public person{
    int emp_id;
    string subject;
    public:
    void acceptStaff(){
        cout << "Enter employee ID: " << endl;
        cin >> emp_id;
```

```
    cout << "Enter subject: " << endl;
    cin >> subject;
  }

  void displayStaff(){
      cout << "Name: " << name << endl;
      cout << "Age: " << age << endl;
      cout << "Emp ID: " << emp_id << e
      cout << "Subject taught: " << subje
                                << endl
  }
};

int main(){
    Student s;
    Staff t;

    s.acceptPer();
    s.acceptStud();
    t.acceptPer();
    t.acceptStaff();

    s.displayStud();
    t.displayStaff();

    return 0;
}
```

Q4 Write a program to implement hybrid inheritance.

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

class College{
    protected:
        string cname;
};

class Employee: protected College{
    protected:
    string emp_name;
    int id;
};

class Staff: public Employee{
    string name;
    int deptId;
    public:
    void accept Emp(){
        cout<<"Enter clg. name: "<<endl;
        cin >> cname;
        cout <<"Enter Emp. name: "<<endl;
        cin >> emp_name;
        cout <<"Enter ID: "<<endl;
        cin >> id;
        cout <<Enter staff name: "<<endl;
        cin >> name;
```

```
        cout << "Enter dept id :- " << endl;
        cin >> dept Id;
    }

    void display Emp (){
        cout << "College : " << cname << endl;
        cout << "Emp name : " << emp_name << endl;
        cout << "ID : " << id << endl;
        cout << "staff Name : " << name << endl;
        cout << "Department ID : " << deptId << endl;
    }
};

class Student : protected College {
    string stu_name;
    int roll;
    public :
    void accept Stud(){
        cout << "Enter college name : " << endl;
        cin >> cname;
        cout << "Enter name : " << endl;
        cin >> stu_name;
        cout << "Enter roll number : " << endl;
        cin >> roll;
    }

    void displayStud(){
        cout << "College : " << cname << endl;
        cout << "Student name : " << stu_name << endl;
        cout << " Roll Number : " << roll << endl;
```

```
int main(){
    Staff staff1;
    staff1.acceptEmp();
    staff1.displayEmp();

    Student stud1;
    stud1.acceptStud();
    stud1.displayStud();

    return 0;
}
```

c) WAP to demonstrate virtual base class. Assume suitable data with figures.

```
#include <iostream>
using name space std;

class CollegeStudent{
protected:
    int student_id;
    string ccode;
public :
    void accept (){
        cout << "Enter Student ID: ";
        cin >> student_id;
        cout << " Enter College Code:";
        cin >> ccode;
    }

    void display (){
        cout << "Student ID: " << student_id << endl;
        cout << "College Code : " << ccode << endl;
    }
};

class Test : virtual public CollegeStudent{
protected:
    float percentage;
public :
    void accept (){
```

```
tual base class.
ith figures.

ID: ";

Code: ";

student_id << endl;
ccode << endl;

llegeStudent{
```

```
    CollegeStudent :: accept();
    cout << "Enter Test percentage: ";
    cin >> percentage;
}

void display (){
    CollegeStudent :: display();
    cout << "Test Percentage: " << percentage
                                << " % " << endl;
}
};

class Sports : virtual public CollegeStudent{
protected:
    char grade;
public:
    void accept(){
        cout << "Enter Sports Grade: ";
        cin >> grade;
    }

    void display (){
        cout << "SportsGrade: " << grade
                                << endl;

    }
};
```

```
class Result: public Test, public Sports{
    float tot_marks;
public:
    void accept(){
        CollegeStudent :: accept();
        cout<< "Enter Test Percentage :";
        cin >> percentage;
        cout << " Enter Sports Grade:";
        cin >> grade;
        cout<< " Enter Total Marks:";
        cin >> tot_marks;
    }

    void display (){
        CollegeStudent :: display();
        cout<< "Test Percentage "<< percentage
                << "%" << endl
        cout << " Sports Grade: " << grade << endl
        cout << " Total marks: " << tot_marks << endl
    }
};

int main(){
    Result r;
    cout << "--- Enter Student Details ---" << endl
    r. accept ();
    cout << "\n--- Student Result ---" << endl
    r. display();
    return 0;
}
```

7/10

Experiment - 7

Q1 WAP using fun
the area of
and area o

```
#include <i
using namesp

class Area{
public :
    float c

        re
    }

    float
        }
};

int main (){
    Area
    cout<< "A
    cout<< "Are
```

Experiment - 7

Q1 WAP using function overloading to calculate the area of a laboratory (rectangular) and area of Class room (square)

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

class Area{
public :
    float calculate (float length, float
                                    breadth){
        return length * breadth;
    }

    float calculate( float side){
        return side * side;
    }
};

int main (){
    Area a;
    cout<< "Area of Laboratory : "<< a. calculate(10,6) << endl;
    cout<< "Area of square : "<< a. calculate(5) << endl;
```

Q2 WAP using function overloading to calculate the sum of 5 float values and sum of 10 integer values.

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

class Sum{
public:
    int total(int a[], int n){
        int s=0;
        for(int i=0; i<n; i++)
            s+=a[i];
        return s;
    }

    float total(float a[], int n){
        float s=0;
        for(int i=0; i<n; i++)
            s+=a[i];
        return s;
    }
};

int main(){
    Sum s;
    int marks[10]={45,56,67,78,89,90,
                   76,88,92,85};
    float grades[5]={9.2,8.7,9.5,8.9,9.0
```

```
cout<< " Sum of 10 student marks: "<<
        s.total (marks,10) << endl;
cout<< " Sum of 5 student grade points: "
        << s.total (grades, 5) << endl;

return 0;
}
```

Q3 WAP to demonstrate the Compile Time and implement Unary - operator when used with the object so that the numeric data member of the class is negated

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

class Teacher{
    int experience;
public:
    Teacher (int e){
        experience = e;
    }

    void display(){
        cout<< " Experience: "<< experience<<"
                "year"<<endl;
    }
```

```
    void operator -(){
        experience = -experience;
    }
};

int main(){
    Teacher t1(10);
    t1.display();
    -t1;
    cout<<"After negation:";
    t1.display();
    return0;
}
```

Q4 WAP to implement the Unary++ operator when used with the object so that the numeric data member of the class is Incremented.

```
#include <iostream>
using name space std;

class Student{
    int count;
public:
    student (int c=0){
        count = c;
    }
```

```
void operator++(){
    ++count;
}

void operator++(int){
    count++;
}

void display{
    cout << "Student count: " << count << endl;
}
};

int main(){
    Student s1(50);
    cout << "Before increment: " << endl;
    s1.display();

    ++s1;
    cout << "After pre-increment: " << endl;
    s1.display();

    s1++;
    cout << "After post-increment: " << endl;
    s1.display();
    return 0;
}
```

Experiment - 8

Q1 WAP to overload the '+' operator so that 2 strings can be concatenated.

```
#include <iostreams
#include <strings>
using namespace std;

class Combine {
    string str;
public:
    Combine (string s= "") {
        str = s;
    }

    Combine operator+ (Combic &obj){
        return Combine (str+ obj .str);
    }

    void display (){
        cout << str << endl;
    }
};

int main (){
    combine s1("xyz"), s2("pqr"), s3;
    s3 = s1+s2;
    cout << "Concatenated string:";
    s3 . display ();
}
```

Q2 Write a pro
Elogin havi
word. Decla
Derive Em
class from
details and
employee

```
# include
# include <
using name

class I log
protected:
    string
public:
    virtual
        cout
        cin
        cou
        cin
    }

    virtu

    }
};
```

Q2 Write a program to create a base class ILogin having data members name and password. Declar accept() function virtual. Derive EmailLoging and MembershipLogin class from ILogin. Display Email Login details and member ship login details of the employee

```
#include <iostream>
#include <string>
using name space 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 << name<<endl;
        cout << "Password: "<< password << endl;
    }
};
```

```
class EmailLogin : public I login {
    string email;
public :
    void accept() override {
        cout<<"Enter Email ID: ";
        cin >> email;
        Ilogin::accept();
    }

    void display () override {
        cout<<"\n --- EmailLogin Details ---"
                                <<endl;
        cout<<"Email ID: "<< email<<endl;
        I login :: display ();
    }
};

class Membership login : public ILogin{
    string membership ID;
public :
    void accept() overide{
        cout<<"Enter Membership ID: ";
        cin >> membership ID.
        I Login::accept();
    }

    void display() override {
        cout << "\n -- Membership loginDetails
                    --- " << endl;
```

: login {

, {

l ID: ";

e {

ogin Details --- "

<< endl;

<< email << endl;

blic I login {

ship ID: ";

login Details

endl;

```
    cout << "Membership ID: " << membership ID << 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;
}
```

7/11

Experiment - 9

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

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

int main(){
    ifstream inFile("First.txt");
    ofstream outFile("Second.text");

    if(!inFile){
        cout<< "Error opening First.txt"<<endl;
        return 1;
    }

    char ch;
    while (inFile.get(ch)){
        outFile.put(ch);
    }

    cout<< "File copied successfully"<<endl;

    inFile.close();
    outFile.close();
    return0;
}
```

Q2 Write a program using Fi

```
#includ
#includ
using e

int ma
    if
    if

}

c
i

w

}
```

Q2 ~~Write~~ WAP to Count Digits and Spaces using File Handling

```
#include <iostream>
#include <fstream>
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 Handl

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

int main(){
    ifstream file ("First.txt");
    if(!file){
        cout<<"Error"<< endl;
        return1;
    }

    string word;
    int count =0;

    while(file>> word)
        count ++;

    cout<< "Total words: "<< count<< endl;
    file.close();
    return0;
}
```

Q4 Write a C++ program to Count Occurence of a given Word using File Handling

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

int main(){
    ifstream file("first.txt");
    if (!file){
        cout<<"Error"<<endl;
        return 1;
    }

    string word, target;
    int count=0;

    cout<<"Enter word to count:"<<endl;
    cin>>target;

    while (file>>word){
        if (word= target)
            count++;
    }

    cout<<"Occurence of \""<<target<<"\":"
                <<count<<endl;

    file.close();
}
```

Experiment - 10

Q1 WAP to find Sum of array of elements using function template.

```
#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 iArr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    float fArr[10] = {1.1, 2.2, 3.3, 4.4, 5.5, 1.1, 2.2, 3.3, 4.4
    double dArr [10] = {0.5, 1.5, 2.5, 3.5, 4.5, 0.5
                        1.5, 2.5, 3.5, 4.5};

    cout << "Sum of Integer array: " << sumArray (iArr
                                        10) <<
    cout << "Sum of float array: " << sumArray (fArr
                                        10) <<
    cout << "Sum of Double array: " << sumArray (dA
                                        10) <<
    return 0;
}
```

Q2 WAP to write a Square Function using template specialization

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

template <typename T>
T square(T num){
    return num* num;
}

template<>
string square<string>(string str){
        return str+str;
}

int main(){
    int num = 5;
    double dec_num = 2.5;
    string str = "Apple";

    cout<< "Square of integer: "<< square(num)<<endl;
    cout<< "Square of double: "<< square(dec_num)
                                <<endl;

    cout<<" Square of string: "<< square(str)<<endl;

    return 0;
}
```

Q WAP to build a Simple Calculator using Class template:

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

template <class T1, class T2> class Calc{
    public:
    T1 x;
    T2 y;

    Calc (T1 n, T2 m){
        x = n;
        y = m;
    }

    void sum(){
        cout << "Sum: " << x+y << endl;
    }

    void diff(){
        cout << "Difference: " << x-y << endl;
    }

    void pro(){
        cout << "Product: " << x*y << endl;
    }
```

```
int main(){
    Calc<int, int> n(100, 200);

    n.sum();
    n.diff();
    n.prod();
    n.quo();
    n.rem();
    n.power();
    n.min_num();
    n.max_num();
    n.sin_x();
    n.cos_y();
}
```

Experiment - 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;
}
```

Experiment - 12

Q WAP to implement Stack.

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

stack<int> stack1;

void disp(){
    stack<int> temp = stack1;
    while (!temp.empty()){
        cout << temp.top() << " ";
        temp.pop();
    }
    cout << endl;
}

int main(){
    int num;
    cout << "Enter a no. :" << endl;
    cin >> num;

    for (int i=0; i<num; i++){
        int temp;
        cout << "Enter element at position" << i+1
                                            << endl;
        cin >> temp;
        stack1.push(temp);
    }
```

```
cout<< endl;
    cout<<"Top most element:"<<stack.top()
                                    <<endl;

    cout << endl;
    cout<<"Stack elements (top to bottom)--->"
                                    << endl;

    disp();
    cout<<endl;
    cout<<"Pop Function:"<<endl;
    stack1.pop();
    disp();
    stack1.pop();
    disp();
    stack1.pop();
    disp();

}
```

Q WAP to implement STL Queue

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

int main(){
    queue<int>q;
    for(int i=0; i<=10; i++){
        q.push(i*10);
    }

    cout << endl;
    cout<<"front element"<<q.front()<<endl;
    cout<<"Back element"<<q.back()<<endl;

    cout << endl;
    q.pop();
    cout << "After one pop, front="<<
                                q.front()<<
                                endl;

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

Q

Experiment -11

Q WAP to implement vector 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 << "Intial vector:" << endl;

    vector <int>:: iterator I;

    for (I = v.begin(); I != v.end(); ++I)
    {   cout << *I << " " << endl;
    }

    cout << "Multiply by 10" << endl;
    for (I = v.begin(); I != v.end(); ++I){
        *I = (*I) * 10;
    }

    cout << "New Vector :" << endl;
    for (I = v.begin(); I != v.end(); ++I){
        cout << *I << " " << endl;
    }
}
```

7/11