PAGE No.
DATE / /

OOP Experiment - 1

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

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

class Student {
    string name;
    int roll_no;

public:
    void accept () {
        cout << "ENTER STUDENT NAME :";
        cin >> name;
        cout << "ENTER ROLL NO :";
        cin >> roll_no;
    }

    void display () {
        cout << "STUDENT NAME :" << name
        cout << "ROLL NO :" << end l;
                << roll_no << end l;
    }
};

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

Page No.
Date : / /

Output :-

ENTER STUDENT NAME - VARUN
ENTER ROLL NO - 43.

Student details :
Name : Jaya + Varun
Roll no : 43.

PAGE No.
DATE / /

(2) WAP to declare a class book having data members as id, name, price. Accept data for 2 books & display data of book having greatest 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;
    }

    float getPrice () {
        return price;
    }
};
```

```
int main () {
    int n;
    cout << "Enter number of book :";
    cin >> n;

    Book books [n];

    for (int i = 0; i <n ; i++ ) {
    cout <<"\n Enter details of Books"
         << i+1 << ":" << end end];
    books [i] . accept ();
}
    int maxIndex = 0;
    for (int i=1 ; i <n; i++)
    if (books [i] . getPrice () > books
        [maxIndex] . getPrice () ) {
    }
}
    cout << "\n Book with greatest
         Price :" << end1 ;
    books [maxIndex] . display ();
    return 0;
}
```

PAGE No.
DATE / /

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
——— " ——— : tdhj .
——— " ——— : vjp .
——— " ——— : 65 .

PAGE No.
DATE / /

③ 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 {
    int H, M, S;

public:
    void accept () {
        cout << "Enter hours :" ;
        cin >> H;
        cout << " Enter minutes :" ;
        cin >> M;
        cout << "Enter seconds :" ;
        cin >> S;
    }

    void displayTotalSeconds () {
        int totalseconds = H * 3600 + M
                           * 60 + S;
        cout << " Total time in second:"
             << totalSeconds << endl;
    }
};

int main () {
    Time t;
    t.accept ();
    t.displayTotalSeconds ();
    return 0;
}
```

PAGE No.
DATE / /

Output :-

Enter hours : 4.
Enter minutes : 23.
Enter seconds : 54.
Total time in seconds : 15834.

15/7

PAGE No.
DATE / /

OOP Experiment - (2)

(1) WAP to declare a class 'city' having data members as name & population. Accept this data for 5 cities & display name of city having largest population.

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

class city {
public :
String name ;
  int P ;
  void accept ()
{
cout <<" enter city name & population" ;
cin >> name >> P
}
};

int main ()
{
city c[5];
for (int i=0; i<5 ; i++)
{
c[i]. accept ();
}

int maxIndex =0;
for (int i = 1; i<5; i++){
  if ( c[i] . P > c [maxIndex] . P) {
    maxIndex =i ;
}
```

```
}
cout << " city with highest population : " << c[max
        Index]. name << endl ;
return 0;
}
```

(2) WAP to declare a class account having data members as account no. & 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 {
    int acc_no;
    float balance;
public:
    void getdata () {
    cout << "Enter account number : ";
    cin >> acc_no;
    cout << "Enter balance : ";
    cin >> balance;
}

    void applyInterest () {
    if (balance >= 5000) {
    cout << " account number : " <<
```

Vasad Patel

VJP ☆

PAGE No.
DATE / /

```
balance += balance * 0.10;
    }
}

void display() {
    if (balance >= 5000) {
    cout << "Account Number: " << balance << endl;
    << "Balance with interest: " << balance << endl;
    }
}

int main() {
    Account acc[10];
    cout << "enter details 10 accounts
        : \n";

    for (int i=0; i<10; i++) {
    cout << "\n Account " << i+1 << ":\n";
    acc[i].getData();
    }
    cout << "\n Accounts with balance>
                = 5000 after
    adding 10% interest: \n";
    for (int i=0; i<10; i++) {
    acc[i].applyInterest();
    acc[i].display();
    }
    return 0;
}
```

PAGE No.
DATE / /

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

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

void getData () {
    cout << " enter name : " ;
    cin >> name ;
    cout << " enter post : " ;
    cin >> post ;
}
void showHOD () {
IF (post == 'HOD' || post == 'hod' )
    cout << 'HOD Name :"
        << endl ;
    }
}
int main () {
    staff s [5];

for (int i = 0 ; i<5 ; i++ ) {
    cout << " In enter details of staff" <<
    i+1 <<" : \n" ;
    s [i] . getData ();
```

```
}
cout << "\n List of staff who are HOD:\n";
for (int i = 0; i < s; i++) {
    s[i].showIfHOD();
}

return 0;
}
```

Q1
19/9

PAGE No.
DATE / /

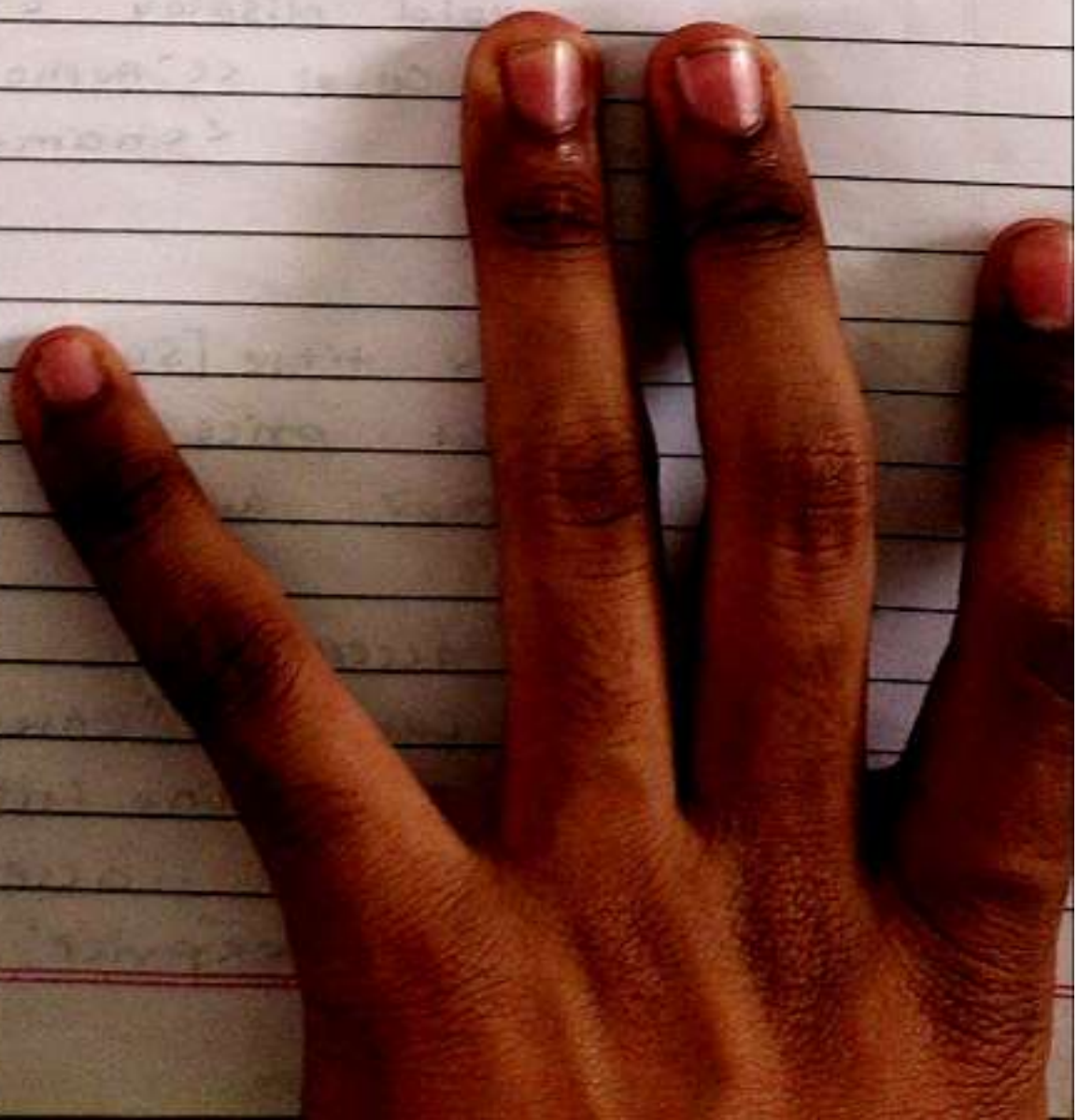
OOP Experiment - (3)

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

```
#include <iostream>
#include <cstdio>
using namespace std;
class Book {
class Author {
public:
    char name[50];
    void accept() {
        cout << "Enter Author Name:";
        cin.getline(name, 50);
    }
    void display() {
        cout << "Author Name:"
             << name << endl;
    }
};
    char title[50];
    float price;
    Author author;
public:
    void accept() {
        cout << "Enter Book Title:";
        cin.getline(title, 50);
        author.accept();
        Cout << "price:" << Price << endl;
```

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

    ptr → accept ();
    ptr → display ();
    return 0;
}
```



PAGE No.
DATE / /

(7) WAP to declare a class student having data members roll no. & percentage using 'this' pointer involve member functions to accept & 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 no: ";
        cin >> this → roll_no;
        cout << "enter percentage :";
        cin >> this → percentage;
    }

    void display () {
    Cout << " in Roll no : " << this → roll_no << endl;
    cout << " percentage :" << this → percentage << "%. " << endl;
    }
};

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

PAGE No.
DATE / /

(8) WAP to demonstrate the use of nested class.

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

class OuterClass {
    int outerData;

public:
    OuterClass(int val) : outerData(val) {}

    class InnerClass {
        int innerData;

public:
        InnerClass(int val) : innerData(val) {}

        void display(OuterClass &outer) {
            cout << "Inner data:" << innerData << endl;
            cout << "Accessing Outer Data from Inner
                Class :" << outer.outerData << endl;
        }
    };
};

int main() {
    OuterClass outer(100);
    OuterClass::InnerClass inner(50);

    inner.display(outer);

    return 0;
}
```

PAGE NO.
DATE / /

Experiment - (4)

(1) WAP to swap two numbers from same class using object as function arguments. Write swap function as member function.

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

Class Number {
  private :
    int vawe :

public :
    Number (int v) { .valve = vi }

void SwapNumber ( Number &n ) {
  int temp = valve ;
  valve =n .valve ;
  n. valve =temp ;
}
    void display () {
    Cout << valve << endl ;
}
};
int main () {
    int main () {
      int a,b ;
      Cout <<" Enter first NO :" ;
      cin >> a ;
      Cout<<" enter second NO :" ;
      cin >> b ;
```

```
Number num1(a), num2(b);
cout << "\n Before swapping :" << endl;
cout << " Num1 :"; num1.display();
cout << " Num2 :"; num2.display();

num1.swapNumbers(num2);
cout << "\n After swapping :" << endl;
cout << " Num1 :"; num1.display();
cout << " Num2 :"; num2.display();

return 0;
}
```

PAGE No.
DATE / /

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

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

class Number {
private:
    int value;

public:
    Number (int v) { value = v; }

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

int main () {
int a, b;
cout << "Enter first no: ";
cin >> a;
cout << "enter 2nd no: ";
cin >> b;

Number num1 (a), nvm2 (b);
cout << "\n Before swapping: " << endl;
cout << " Num1 = "; nvm1.display ();
cout << " Num2 = "; Num2.display ();
```

```
SwapNumber (num1, num2);

cout << "In After Swapping :" << endl;
cout << " Num1 : " ; num1.display();
cout << " Num2 : " ; num2.display();

return 0;
}
```

PAGE No.
DATE / /

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

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

class B;
class A
  private:
    int value A;
  public:
  A (int v) { value A = v; }

    friend void swapNumbers (A &x, B &y);
    void display () { cout << value A << endl; }
};
class B {
  private:
    int value B;

  public:
    B (int v) { value B = v }

  friend void swapNumbers (A &x, B &y);
    void display () { cout << &value B << endl;
};
void swapNumbers (A &x, B &y) {
  int temp = x.value A;
  x.value A = y.value B;
  y.value B = temp;
}
```

```
int main () {
    int a, b;
    cout << " enter no in class A : " ;
    cin >> a;
    cout << " enter no in class B : " ;
    cin >> b;

    A objA (a);
    B objB (b);

    cout << "\n Before swapping : " << endl;
    cout << " A : " ; objA.display ();
    cout << " B : " ; objB.display ();

    swapNumbers (objA , objB );

    cout << " \n After swapping : " << endl;
    cout << " A " ; objA.display ();
    cout << " B " ; objB.display ();

    return 0;
}
```

PAGE No.
DATE / /

(4) WAP to create Result of Rem which stores the marks of one student. Read the value of 2 marks for both the class objects & compute the average of the results.

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

class Result2;

class Result1 {
private:
    float marks1;

public:
    void readMarks() {
        cout << "Enter marks for result 1:";
        cin >> marks1;
    }
    friend void computeAverage(Result1 r1, Result2 r2);
};

class Result2 {
private:
    float marks2;

public:
    void readMarks() {
        cout << "Enter marks for result 2:";
        cin >> marks2;
    }
```

```
friend void computeAurage (Result1 r1, Result2 r2);
};
void computeAverage (Result r1, Result2 r2) {
    float avg = (r1.marks1 + r2.marks2) / 2;
    cout << "Average marks : " << avg << endl;
}
int main () {
Result1 obj1;
Result2 obj2;

obj1.readmarks ();
obj2.readmarks ();

computeAurage (obj1, obj2);

return 0;
}
```

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

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

    class B;
    class A {
    private
        int numA;

    public :
        A(int n) { numA = n; }
        friend void findGreatest (A a, B b);
};
    class {
    private
        int numB;

    public :
    B (int n) { numB = n; }
        friend void findGreatest (A a, B b);
};
    void findGreatest (A a , B b) {
        if (a.numA > b.numB)
            cout << "Greatest No: " << a.numA << endl;
    else if (b.numB > a.numA)
        cout << "Greatest no: " << b.numb << endl;
    else
        cout << "Both no.s are equal. " << a.numA << endl;
}
```

```
int main() {
  int x, y;
  cout << " Enter number for class A: ";
  cin >> x;
  cout << " Enter no. for class B: ";
  cin >> y;

  A objA(x);
  B objB(y);

  findGreatest (objA, objB);
  return 0;
}
```

PAGE No.
DATE / /

⑤ Create two classes .Class A & classB each with a private integer. write a friend function Sum that access private data from both from both classes xtrn the sum.

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

class classB;
class classA {
private:
    int numA;
public:
    classA(int a) {
    numA = a;
    }
    friend int sum (classA, classB);
};

class classB {
private:
    int numB;
public:
    classB(int b) {
    numB = b;
    }
};

int sum (classA objA, classB objB) {
    return objA.numA + objB.numB;
}
```

PAGE No.
DATE / /

```
int main () {
    int a, b ;
    cout << " Enter first no (class A) : " ;
    cin >> a ;
    cout << " Enter second no (class B) : " ;
    cin >> b ;

    ClassA objA(a) ;
    ClassB objB (b) ;

cout << " Sum : " << Sum (objA, objB) << endl ;
    return 0 ;
}
```

PAGE No.
DATE / /

② WAP a class number that contains a private integer. use a friend function SwapNumbers to swap the private values of two Numbers objects.

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

class Number {
  private:
    int value;
  public:
    Number (int v) {
      value = v;
    }
    void display () {
      cout << value;
    }
    friend void SwapNumber (Number &n1, Number &n2); };
};
void SwapNumbers (Number &n1, Number &n2);
{
  int temp = n1.value;
  n1.value = n2.value;
  n2.value = temp;
}
int main () {
int a, b;
cout << "enter first no:";
cin >> a;
```

cout << "enter second no :";
cin >> b;

Number num1(a), num2(b);

cout << "In Before Swapping :" << endl;
cout << " num1 =";
num1.display();
cout << " num2 =";
num2.display();

SwapNumbers (num1, num2);

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

return 0;
}

PAGE No.
DATE / /

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

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

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

  public:
    cube (int v)
    friend void findGreater (box, cube);
};
void findGreater (Box b, cube c) {
  if (b.volume > c.volume) {
  cout << "box has larger volume" << b.volume
  << endl;
```

```
} else {
        cout << "Both have the same volume"
             << b.volume << endl;
    }
}

int main () {
    int boxVol , cubeVol;
    cout << "Enter volume of box: ";
    cin >> boxVol;
    cout << "enter volume of cube: ";
    cin >> cubeVol ;

Box b (boxVol);
Cube c (cubeVol);
FindGreater (b, c);

return 0;
}
```

PAGE No.
DATE / /

Create a class complex w/ real and imaginary parts as private members, use a friend function to add two complex nos. and return as a new complex object

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

class Complex {
  private:
    float real;
    float imag;

  public:
    Complex (float r=0, float i=0) {
      real = r;
      imag = i;
    }
    friend Complex addComplex (Complex, Complex);
    void display () {
      cout << real << " , " << imag << " i " << endl;
    }
};
Complex addComplex (Complex c1, Complex c2) {
    Complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c.2imag;

    return result;
}
```

```
int main() {
    float r1, i1, r2, i2;
cout << "Enter real & imaginary part of
  first complex no :";
cin >> r1 >> i1;
cout << "Enter real & imaging part of
  Second complex no:";
cin >> r2 >> i2;

Complex c1 (r1, i1);
Complex c2 (r2, i2);

Complex sum = add Complex ((1, c2);
  cout << "Sum of Complex numbers:";
  sum . display ();
  return 0;
}
```

Create two classes : BankAccount and Audit
BankAccount stores 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 BankAccount {
private:
    double Balance;

public:
    BankAccount (double b)
    friend void auditAccount (BankAccount);
};
    class Audit {
};
    void auditAccount (BankAccount acc ) {
    cout << "Balance : " << acc.balance << endl;
}
    int main () {
    double bal;
    cout << "Enter account balance :" ;
    cin >> bal;

BankAccount account (bal);
auditAccount (account);
        return 0;
}
```

PAGE No.
DATE / /

Experiment 07

(a)

```
#include <iostream>
using namespace std;
class NumberPrinter {
public:
    NumberPrinter(int n) {
        cout << "Numbers from 1 to " << n << ":" << endl;
        for (int i = 1; i <= n; i++) {
            cout << i << " ";
        }
        cout << endl;
    }
};
int main() {
    int n;
    cout << "Enter the value of n: ";
    cin >> n;
    NumberPrinter np(n);
    return 0;
}
```

Output:

Enter the value of n: 2
Numbers from 1 to 2:
1 2

(2)

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

class student {
private :
string name;
float percentage;

public :
student (string n, float p) {
    name = n;
    percentage = p;
}
    void display () {
cout << "student Name :" <<name << endl;
cout << "percentage :" <<percentage<< "%" <<endl;
};
}

int main () {
sring name;
float percentage;

cout << " Enter student name :" ;
getline (:, n ,name);

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

Student s (name, percentage);

cout << "In student details: In";
s.display ();

return 0;
}

output
Enter student name: Varad
Enter percentage : 87

Student details:
Student Name: Varad
percentage : 87 %

2.3 #include <iostream>
using namespace std;

class college {
private:
int roll-no;
string name;
string course;

public:
college (int r, string n, string c
: "Computer Engineering") {

roll-no = r;
name = n;
course = c;
}

```
void display(){
 cout << " Roll NO :" << roll_no << endl;
 cout << " Name :" << name << endl;
 cout << " Course :" << course << endl << endl;
};

int main(){
    int roll;
    String name, course;
cout << " Enter details for student 1 :" << endl;
cout << " Roll NO :";
    cin >> roll;
    cin.ignore();
    cout << " Name :";
    getline(cin, name);
    cout << " course :";
    getline(cin, course);
    college student 2(roll, name, course);
cout << " In student Details \n";
    Student 1.display();
    Student 2.display();
    return 0;
}
```

PAGE No.
DATE / /

Ouipvt :
Roll Number : 37
Name : varad
Course : computer engineering
Roll Number : 18
Name : Sonam
Course : computer Engineering

Q.4) WAP to demonstrate Constructor over loading

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

class Student {
  int roll;
  string name;
  public:
  Student () {
  name = "unknown";
  roll =0;
}
  Student (String n) {
    name = n;
    roll = 0;
}
  Student (string , int r) {
    name =n;
    roll = r;
  void display () {
    cout<<" name :"<< name <<endl;
    cout<<" RollNumber :"<< roll <<endl;
```

```
    }
}
    Student (string n){
    roll = "Unknown";
}
    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 ("Varad");
Student S3 ("Sonam", 32);

S1.display();
S2.display();
S3.display();
return 0;
}
```

PAGE No.
DATE / /

OUTPUT:

Name : unknown
Roll Number : 0
Name : Varad
Roll No 0
Name : Soham
Roll number : 18

PAGE No.
DATE / /

* Experiment 6 *

Q.1) 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 >> name;
    cout << " Enter marks1 : " << endl;
    cin >> m1;
    cout << Enter marks 2 : " << endl;
    cin >> m2;
```

```
void calwrare ()
    Pri per : (m1+m2) / 2
    Cout << " Department : " << dname << endl;
    cout << " Name : " << sname << endl;
    Cout << " Percentage : " << per << endl;
};
int main () {
    marks m;
    m. accept ();
    m. calwrare ();

    return 0;
}
```

Output:

Enter department :
Varied CSE
Enter marks 1 ;
88
Enter marks 2;
91
Department : CSE
Name : Varied
Percentage : 89

PAGE No.
DATE / /

Q.2) 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 marks 1 :"<<endl;
    cin>> m1;
    cout<<" ENter marks 2 :"<<endl;
    cin>> m2;
```

```
}
void calculate ()
{
per = (m1+m2)/2;
cout << " Department : " << aname << endl;
cout << " Name : " << sname << endl;
cout << " percentage : " << per << endl;
}
};
int main()
{
marks m;
m.accept();
m.calculate();
}
```

Q.3) WAP to implement hierarchial inheritance

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

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

```
class Student : public Person{
    int roll;
    float per;
    public :
    void acceptstd (){
        cout << " Enter roll number\n";
        cin >> roll;
        cout << "Enter Percentage:" << endl;
        cin >> per;
    }
    void displaystd(){
        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 << " EmpID : " << emp_id << endl;
    cout << " Subject taught : " << subject << endl;
```

```
};
};

int main (){
Student s;
Staff t;

s.acceptPer();
s.acceptStd();
t.acceptPer();
t.acceptStaff();
s.displayStd();
t.displayStaff();

return 0;
}
```

Q.4) WAP to implement hybrid inheritance.

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

class College {
    protected:
    string cname;
};

class Employee : protected College {
    protected
    String emp_name;
    int id;
};
```

PAGE NO.
DATE / /

```
class Staff : public Employee {
    string name;
    int deptId;
    public:
void acceptEmp() {
cout<<" Enter eg. name : "<<endl;
cin >> ename;
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 >> deptID;
}
    void displayEmp() {
cout <<" College : "<< ename <<endl;
cout <<" Emp name : "<<emp-name<<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 display stud() {
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;
}

PAGE No.
DATE / /

* Experiment - 7 *

Q.1 OOP using function over loading 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 breath){
        return length * breath ;
  }
        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;
```

PAGE No.
DATE / /

11.2] WAP using function overloading to calculate the sum of 5 float values and sum of 10 integers 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, 83 } ;
    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;
}

(8) WAP to demostrate the compile time and implement unary operator when used with the object so that the numeric data member of class is negated.

#include <iostream>
using namespace std;

class Teacher {
public experience;
}
void display() {
cout << "Experience :" << experience << "year" << endl;
}

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

int main() {
Teacher t1(10);
t1.display();
-t1;
cout<<"After negation";
t1.display();
return 0;
}
```

Q4) WAP to implement the unary -- operator when used with one object so that the numeric data member of one class is decremented.

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

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

```
void operator ++ (){
    ++ count ;
}
    void operator ++ (int ){
        count ++;
    }
    void display {
        cout <<" Student count :" << count <<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;
}
```

PAGE No.
DATE / /

* Experiment - 8 *

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

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

class Combine {
  string str;
public:
  Combine (string s = "") {
      string = s;
  }
Combine operator + (Combine &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 ();
}
```

PAGE No.
DATE / /

Q 2)

```
#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 << name << endl;
    cout <<"Password :" << password << endl;
    }
};
class Email Login : public I login {
    string email;
public:
    void accept () override {
    cout <<" Enter Email ID :";
    cin >> email;
    Ilogin :: accept ();
    }
    void display () override {
    cout << "In ---- Email Login Details---" << endl;
    cout <<"Email ID :" << email << endl;
    Ilogin :: display ();
};
```

```
e <iostream>
e <string>
mespace std;
 I login {
ic:
 name, password;

 void accept() {
<<"Enter name:";
> name;
<"Enter password:";
 password;

 void display() {
"Name:" << name << name << endl;
Password:" << password << endl;
```

```
login : public I login {
mail;

 override {
Enter Email ID:";
mail;
accept();

ay() override {
---- Email login details ---" << endl;
 ID:" << email << endl;
display();
```

```
class membership login : public I login { public: I login {
    string memberID;
public:
    void accept() override {
    cout << "Enter membership ID:";
    cin >> memberID;
    I login :: accept();
    }
    void display() override {
    cout << "---- membership login Details
                                    ----" << endl;
    cout << "membership ID:" memberID
    I login :: display();
    }
};

int main() {
I login * login;

Email login e;
membershipLogin m;

login = &e;
login → accept();
login → display();

login = &m;
login → accept();
login → display();

return 0;
}
```

PAGE NO.
DATE / /

* Experiment - 9 *

(a)

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

int main (){
  ifstream fin ("first.txt", ios::in);
  ofstream fout ("second.txt", ios::);

  if (!fin) {
    cout << "Error: first.txt not found!";
    return 0;
  }
  char ch;
  while (fin.get(ch)){
    fout.put (ch);
  }
  fin.close ();
  fout.close ();

  cout << "File copied successfully!");
  return 0;
}
```

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

int main () {
ifstream fin ("first.txt");
if (!fin) {
cout << "File not found!";
return 0;
}
char ch;
int digits = 0, spaces = 0;

while (fin.get(ch)) {
if (isdigit (ch)) digits++;
if (ch == ' ') spaces++;
}
fin.close();

cout << "Digits : " << digits << endl;
cout << " Spaces: " << spaces << endl;

return 0;
}
```

12/11

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

int main () {
    ifstream fin ("first.txt");
    if (!fin) {
        cout << "file not found!";
        return 0;
    }
    string word;
    int count = 0;
    while (fin >> word) {
        count ++;
    }
    fin.close ();
    cout << "Total words: " << count;
    return 0;
}
```

(d) #incl

(a)

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
    int main () {

    ifstream fin ("first.txt");
    if (!fin) {
     cout << "File not found!";
     return 0;
}
    string word, target;
    int count =0;

    cout << "Enter word to search:");
    cin >> target;

    while (fin >> word) {
     if (word == target) count++;
}
    fin.close ();
cout << "Occurrences of \" << target << "\":
    << count;

    return 0;
}
```

PAGE No.
DATE / /

*Experiment - 10 *

(a)
```
#include <iostream>
using namespace std;
template <class T>
T arraySum (T arr [], int n) {
  T sum = 0;
  for (int i = 0; i < n; i++) {
    sum += arr [i];
  } return sum;
}
int main () {
  int a[10] = {1,2,3,4,5,6,7,8,9,10};
  float b[b] = {1.1, 2.2, 3.3, 4.4, 5.5, 1.1, 2.2, 3.3, 4.4, 5.5};
  double c [10] = {1,2,3,4,5,6,7,8,9,10}
  cout << "Sum of int array:" << arraySum(a,10) << endl;
  cout << "Sum of float array:" << arraySum(b,10) << endl;
  cout << "Sum of double array:" << arraySum(c,10) << endl;
  return 0;
}
```

(b)
```
#include <iostream>
#include <string>
using namespace std;
template <class T>
T squarefunc (T x) {
  return x * x;
}
  template <>
string squarefunc <string> (string s) {
    return s + s;
}
```

```
int main () {
 int n = 5;
String str = "Hello";
cout << " Square of integer :" <<Squarefunc(n)<<endl;
Cout << "Squar of string :" <<Squarefunc(str)<< endl;
   return 0;
}
```

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

temprure <class T>
class Calculator {
  T a, b;
Public:
  Calculator (T x, T y) {
     a = x;
     b = y;
  }
  T add () { return a+b; }
  T Sub () { return a-b; }
  T mul () { return a*b; }
  T div () { return a/b; }
};
int main () {
Calculator <int> c(20, 5);
```

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

template < class T >
class Stack {
    T Stack [100];
    int top;
Public :
 Stack () { top = -1 ; }
 void push (T x) {
 if (top == 99)
    cout <<"Stack overflow \n" };
 else
    cout << " Popped : "<< stack [top--] << endl;
}
 void display () {
 if (top == -1) {
  cout << " Stack is empty \n" ;
  return 0;
}
 for (int i = top ; i >=0 ; i--)
  cout << stack [i] << " " ;
  cout << end l;
 }
};
int main () {
 Stack <int > s;
 S. push (10);
 S. push (20);
 S. push (30);
 S. display ();
 S. pop ();
 S. pop ();
 S. display ();
 return 0;
}
```

PAGE No.
DATE / /

* Experiment - 11 *

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

template <class T>
class vector {
T * arr;
int size;
Public :
    vector (int s) {
    size : s;
    arr : new T [size];
}
    void createvector () {
    cout <<" Enter" << size << " elements : \n" ;
    for (int i=0 ; i<size ; i++ ) {
    cin >> arr [i];
    }
}
    void modify (int index, T vave ) {
    if (index >=0 && index <size) {
    arr [index] : value;
    } else {
    cout <<" Invalid index ! \n" ;
    }
}
    void display () {
    cout <<" ( " ;
    for (int i =0; i < size ; i++ ) {
    cout << arr [i];
    if ( i != size -1 ) cout <<" , " ;
}
```

```
    cout << ")\n";
}
    ~vector () {
        delete [] arr;
};};
    int main () {
    int n;
    cout << " Enter size of vector : ";
    cin >> n;

vector <int > v (n);
v. createVector ();

int index, value;
cout << "Enter index and new value to modify :";
cin >> index >> value;
 v. modify (index, value);

int scalar;
cout << " Enter scalar value to multiply : ";
cin >> scalar;
v. multiply (scalar);

cout << " Final vector : ";
 v. display ();

 return 0;
}
```

12!!

Experiment - 12

(a)

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
stack <int> s
int choice, value;
do {
    cout << "\n-- STACK MENU ---\n";
    cout << "1. Push\n 2. Pop\n 3. Top\n 4. Display\n 5. Exit";
    cout << "Enter choice: ";
    cin >> choice;
switch (choice) {
    case 1:
        cout << "Enter value to push: ";
        cin >> value;
        s.push(value);
        break;
        case 2:
        if (!s.empty()) {
        cout << "Popped: " << s.top() << endl;
        s.pop();
} else {
        cout << "Stack is empty!\n";
} break;
case 4:
    if (s.empty()) {
    cout << "Stack is empty!\n";
} else {
    cout << "stack elements (top to Bottom): ";
    while (!temp.empty()) {
    cout << temp.top() << " ";
```

```
    } cout << endl;
  } break;
} while (choice != 5);
  return 0;
}
```

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

int main() {
  queue <int> q;
  int choice, value;
  do {
    cout << "\n--- QUEUE MENUE ---\n";
    cout << "1. Enqueue \n2. Dequeue\n3. Front \n4. Display
         \n5. Exit";
    cout << "Enter choice :";
    cin >> choice;
  switch (choice) {
    case 1:
    cout << "Enter value to enqueue :";
    cin >> value;
    q.push (value);
    break;
    case 2:
    if (!q.empty ()) {
    cout << "Dequed :" << q.front () << endl;
    q.pop ();
  } else {
      cout << "Queue is empty !\n";
```

```
}
  break;
case 3:
  if (!q.empty())
    cout << "Front element: " << q.front() << endl;
  else
    cout << "Queue is empty!\n";
  break;
case 4:
  if (q.empty()) {
    cout << "Queue is empty!\n";
  } else {
    cout << "Queue elements (front to rear): ";
    queue<int> temp = q;
    while (!temp.empty()) {
      cout << temp.front() << " ";
      temp.pop();
    }
    cout << endl;
  }
  break;
} while (choice != 5);

return 0;
}
```

Run
fail

(b)

```
#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;
    vector <int> :: iterator ),

for (i=v.begin(); i != v.end(); ++(){
      cout << *I << " endl ;
}
cout<<" Multiply by 10" <<endl;
for (i=v.begin() ; i != v.end(); ++(') {
    cout <<* I <<" <<endl;
}
}
return 0;
```