16/7/25 Experiment No. 1

① Student

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

class student {
    public:
        int roll_no;
        char name[50];

        void accept(){
        cout<<" Enter roll number: ";
        cin >> roll_no;
        cout<<" Enter name: ";
        cin >> name;
        }
        void display(){
        cout<<" Roll Number: "<<roll_no<<endl;
        cout<<" Name: "<<name<<endl;
        }
};
    int main()
{
    student s1;
    s1.accept();
    s1.display();
    return 0;
}
```

② Book

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

class Book{
    public:
        char book_name[50];
        int pgs;
        int price;

        void accept(){
        cout<<"Enter name of book:";
        cin>>book_name;
        cout<<"Enter pgs :";
        cin>>pgs;
        cout<<"Enter price:";
        cin>>price;
        }
};
int main(){
    Book b1, b2;
    b1.accept();
    b2.accept();
if(b1.price>b2.price){
  cout<<"The first book has greater price"<<b1.price<<endl;
  }
else if (b2.price>b1.price){
  cout<<"The second book has greater price"<<b2.price<<endl;
}
else{ cout<<"Both have same price."<<endl;
} return 0;
}
```

③ Time:

```
#include<iostream>
~~int main~~ using namespace std;

class Time{
    int hours, mins, secs;
    public:
    void input(){
        cout << "Enter time (HH:MM:SS)";
        cin >> hours >> mins >> secs;
    }
    int totalsecs(){
        return hours * 3600 + mins * 60 + secs;
    }
    void displaytotalsecs(){
        cout << "Total secs =" << totalsecs() << endl;
    }
};

int main(){
    Time t;
    t.input();
    t.displaytotalsecs();
    return 0;
}
```

24/7/25

16/7/25 Experiment No. 2:

① WAP to declare class 'city' having data members as name and population. Accept the data for 5 cities and display name of book having greater price.

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

class city{
public:
        string city_name;
        int population;

        void accept(){
        cout<<"Enter name of the city:";
        cin>>city_name;
        cout<<"Enter population:";
        cin>>population;
    }

        void display(){
        cout<<"City having highest pop:"<<population<<endl;
    }
};
  int main(){
     city a[5];
     for(int i=0; i<5; i++){
        a[i].accept();
    }
     int max=0;
     for(int i=1; i<5; i++){
        if(a[i].population>a[max].population){
```

```
            max = i;
        }
    }
    a[max].display();
}
```

(2) WAP to declare a class Account having data members as account_no, balance. Accept data for 5 members and give interest of 10% to acc having balance greater than 5000.

```
#include<iostream>
using namespace std;
class Account{
    public:
    int account_no;
    float balance;
    void accept(){
        cout<<"Enter Account No:";
        cin>>account_no;
        cout<<"Enter balance:";
        cin>>balance;
    }
    void display(){
        cout<<"Account No:"<<account_no<<endl;
        cout<<"Balance:"<<balance<<endl;
    }
};
int main(){
        Account a[5];
    for(int i=0; i<5; i++){
        a[i].accept();

        if(a[i].balance>=5000){
            a[i].balance+=a[i].balance*0.10;
        }
    }
    cout<<"Accounts with updated:\n";
    for(int i=0; i<5, i++){
```

```
        a[i].display();
    }
    return 0;
}
```

Staff

③ WAP to declare a class having data members name and post. Accept the data for 5 employees and display the name of the person having post HOD.

```
#include<iostream>
#include<string.h>
using namespace std;
class Staff{
    public:
        char name[50], post[50];
    void accept(){
        cout<<"Enter name of employee";
        cin>>name;
        cout<<"Enter the post:";
        cin>>post;
    }
    void display(){
        cout<<"Member having post HOD:"<<name<<endl;
    }
};
int main(){
    Staff a[5];
    int i;
    for(i=0; i<5; i++){
        a[i].accept();
    }
    for(i=0; i<5; i++){
        if(strcmp(a[i].post,"HOD")==0){
            a[i].display;
    }
}
}
```

30/7/25

30/7/25

Experiment No-3

C.W.

a) WAP to declare a class 'book' containing data members ~~roll_no and per~~ book title, author and pri~~ce~~. ~~Using this pointer~~ invoke Accept and display this data for one obj.

```
#include <iostream>
using namespace std;
class book{
        string book_title;
        string author_name;
        int price;
        void accept(){
            cout<<"Enter title:"<<endl;
            cin>>this->book_title;
            cout<<"Enter author name"<<endl;
            cin>>this->author_name;
            cout<<"Enter price:"<<endl
            cin>>this->price;
        }
        void display(){
            this->accept();
            cout<<"The book title:"<<this->book_title<<co
            cout<<"The author:"<<this->author_name<<endl
            cout<<"The price:"<<this->price<<endl;
        }
};
int main(){
        book b;
        book *p;
        p=&b;
        p->display();
}
```

b) WAP to declare a class

```
#include <iostream>
#include <string>
using namespace std;
class student {
public:
    int roll_no;
    float perc;
    void accept() {
        cout<<"Enter roll no: "<<endl;
        cin>>this->roll_no;
        cout<<"Enter percentage: "<<endl;
        cin>>this->perc;
    }
    void display() {
        this->accept();
        cout<<"roll no of student: "<<this->roll_no<<endl;
        cout<<"percentage of student: "<<this->perc<<endl;
    }
};
int main() {
    student s;
    s.display();
}
```

c) WAP to demonstrate nested class.

```
#include<iostream>
using namespace std;
class student {
public:
    class student marks {
    public:
        int roll_no;
        char name[50];
        int m1;
        int m2;

        void accept(){
        cout<<"Enter roll no:";
        cin>>roll_no;
        cout<<"Enter name:";
        cin>>name;
        cout<<"Enter marks for m1:";
        cin>>m1;
        cout<<"Enter marks for m2:";
        cin>>m2;
        }
        void display(){
        cout<<"The roll no:"<<roll_no<<endl;
        cout<<"The name:"<<name<<endl;
        cout<<"Marks for M1 and M2:"<<m1<<m2<<endl;
        }
    };
};
int main(){ student::student marks s;
            s.accept();
            s.display();
}
```

30/7/25

6/8/25

Experiment No 4

1) Swapping using friend function (2 classes)

```
#include<iostream>
using namespace std;
class n2;
class n1{
    int a;
    public:
    void accept(){
        cout<<"Enter ur first number:";
        cin>>a;
    }
    void display(){
        cout<<"First Number ="<<a<<endl;
    }
    friend void swap(n1&s, n2&r);
};
class n2{
    int b;
    public:
    void accept(){
        cout<<"Enter ur second number:"
        cin>>b;
    }
    void display(){
        cout<<" Second number ="<<b<<endl;
    }
    friend void swap(n1&s, n2&r);
};
```

```
void swap(n1 &s, n2 &r){
int t = s.a;
s.a = r.b;
r.b = t;
}
int main(){
n1 s;
n2 r;
s.accept();
r.accept();

cout<<"Before swapping:\n";
s.display();
r.display();

swap(s, r);

cout<<"After swapping:\n";
s.display();
r.display();

}
```

2) Swapping using friend function. (one class)

```
#include <iostream>
using namespace std;
class n1{
    int a, b;
    public:
    friend void accept(){
    cout<<" Enter your numbers: ";
    cin >> a >> b;
  }
    friend void swap(n1& s);
};
void swap(n1& s){
    int t = s.a;
    s.a = s.b;
    s.b = t;
    cout <<" First No: "<<s.a<<endl<<" Second Number "<<
    s.b;
}
int main(){
    n1 s;
    s.accept();
    swap(s);
}
```

3) WAP to create 2 classe Result1 and Result2 which stores the marks of the students. Read the value of marks for both the class objs. and compute avg.

```
#include <iostream>
using namespace std;
class Result2;
class Result1 {
        int m1;
        public:
        void accept(){
        cout<<" Enter your marks for first subject:";
        cin>>m1;
        }
        friend void avg(Result1 &s, Result2 &r);
};
Result1
class Result2{
        int m2;
        public:
        void accept(){
        cout<<" Enter your marks for second sub:";
        cin>>m2;
        }
        friend void avg(Result1 &s, Result2 &r);
};
void avg (Result1 &s, Result2 &r);
float a(s.m1 + r.m2)/2;
cout<<" Averge = "<<a;
}
int main(){
```

R.T.O. →

```
Result1 s;
Result2 r;
s.accept();
r.accept();
avg(s, r);
}
```

4) WAP to find greatest number among 2 numbers from 2 diff classes using friend function.

```
#include <iostream>
using namespace std;
class n2;
class n1{
        int a;
        public:
        void accept(){
        cout<<"Enter first number:";
        cin>>a;
        }
        friend void greatest(n1&s, n2&r);
};
class n2{
        int b;
        public: void accept(){
        cout<<"Enter Second number:";
        cin>>b;
        }
        friend void greatest(n1&s, n2&r);
};
void greates(n1&s, n2&r){
        float g = s.a;
        if(s.a > r.b){
        cout<<"Greater number is:"<<s.a;
        else{
        cout<<"Greater number is:"<<r.b;
int main(){
```

→

```
n1 s;
n2 r;
s.accept();
r.accept();
greatest(s,r);
```

5) Swapping 2 nos. without friend function.

→
```
#include<iostream>
using namespace std;
class Demo{
public:
        int p,q;
        void accept(){
            cout<<" Enter 2 nos:-"<<endl;
            cin>>p>>q;
    }
     void display(){
            cout<<" After swapping:-"<<" Value of p=
                "<<p<<" Value of q="<<q;
    }
    void swap(demo &t){
            int temp=t.p;
            t.p=t.q;
            t.q=temp;
};};
int main(){
    demo k;
    k.accept();
    k.swap(k);
    k.display();
}
```

6) Create 2 classes, class A and class B in a private integer. Write a friend function sum() that can access private ~~da~~ data from both classes.

```
#include <iostream>
using namespace std;
class B;
class A{
        int a;
public:
        void accept1(){
        cout<<"Enter first no: "<<endl;
        cin>>a;
    }
      friend int sum(A p, B q);
};
class B{
    int b;
public:
        void accept2(){
        cout<<"Enter the second no: "<<endl;
        cin>>b;
    }
      friend int sum(A p, B q);
};
int sum(A p, B q){
    int sum;
    sum = p.a + q.b;
    return sum;
}
int main(){
        A k;
        B f;
```

```
k.accept1();
f.accept2();
cout<<" The sum of 2 nos:"<<sum(k,f);
}
```

7) WAP to swap 2 ~~nos~~ private integers of the same class using a friend function.

```
#include<iostream>
using namespace std;
class A{
    int a,b;
public:
    void accept(){
        cout<<"Enter 2 nos:"<<endl;
        cin>>a>>b;
    } friend void swapnumbers(A &t);

    void display(){
        cout<<" value of a:"<<a;
        cout<<" value of b:"<<b;
    } friend void swapnumbers(A &t);
};
void swapnumbers(A &t){
    int temp=t.a;
    t.a=t.b;
    t.b=temp;
}
int main(){
        A k;
        k.accept();
        swapnumbers(k);
```

```
k.display();
}
```

8) WAP with class Box/Cube. using friend function determine which obj. has larger volume.

```
#include <iostream>
using namespace std;
class cube;
class box{
    int l, b, h, v1;
public:
    void accept1(){
        cout<<"Enter dimension of box"<<endl;
        cin>>l>>b>>h;
    }
    friend void greatervolume(box p, cube q);
};
class cube{
    int side, v2;
public:
    void accept2(){
        cout<<"Enter the dimesions of cube:"<<endl;
        cin>>side;
    }
friend void greatervolume(box p, cube q);
};
void greatervolume(box p, cube q){
    p.v1=p.l*p.h*p.b;
    q.v2=q.side*q.side*q;
    if(p.v1>q.v2){
        cout<<"The box having greater volume:"<<p.v1
        <<endl;
```

```
}
else {
    cout << "the box having greater vol is: " << q.v2 << v
}
}
int main() {
    box k;
    cube f;
    k.accept1();
    f.accept2();
    greatervolume(k, f);
}
```

9) WAP for addition of 2 complex nos.

```
#include <iostream>
using namespace std;
class complex {
    int x, y;
public:
    void input() {
        cout << "Enter the real part: ";
        cin >> x;
        cout << "Enter imaginary part of no: ";
        cin >> y;
    }
    void sum(complex &a, complex &b) {
        int sumr = a.x + b.x;
        int sumi = a.y + b.y;
        cout << "Sum = " << sumr << " + " << sumi
             << " i" << endl;
};
```

```
int main(){
        complex a, b, c;
        a.input();
        b.input();
        c.sum(a, b);
}
```

10) WAP to find avg. of 3 marks using friend function

```
#include<iostream>
using namespace std;
class Student {
    string name;
    int m1, m2, m3;
    public:
    void accept(){
        cout<<"Enter your name:";
        cin>>name;
        cout<<"Enter your marks for 3 subs:
        cin>>m1>>m2>>m3;
    }
     friend void calc(Student a);
};
void calc(Student s){
        float avg=(s.m1+s.m2+s.m3)/3;
        cout<<"The average of your marks is:"<<avg;
}
int main(){
    Student s;
    s.accept();
    calc(s);
}
```

11) WAP to ~~add~~ print sum of gamma, beta, alpha using one friend function.

```
#include <iostream>
using namespace std;
class beta;
class gamma;
class alpha{
        int a;
        public:
        void accept(){
            cout<<"Enter alpha value:";
            cin>>a;
        }
        friend void sum(alpha s, beta r, gamma n);
};
class beta{
        int b;
        public:
        void accept(){
            cout<<"Enter your beta value:";
            cin>>b;
        }
        friend void sum(alpha s, beta r, gamma n);
};
class gamma{
        int g;
        public:
        void accept(){
            cout<<"Enter your gamma value:";
            cin>>g;
        }
        friend void sum(alpha s, beta r, gamma n);
};
```

```
void sum (alpha s, beta r, gama n){
    int sum = s.a + r.b + n.g;
    cout<<" The sum of your values: "<<sum;
}
int main(){
        alpha s;
        beta r;
        gamma g;
        s.accept();
        r.accept();
        g.accept();
        sum(s, r, g);
}
```

12) WAP to find distance between 2 co-ordinates. using friend function.

```
#include <iostream>
#include <cmath>
using namespace std;
class Point{
        float x, y;
public:
        void accept(){
            cout<<"Enter x and y coordinates:";
            cin>>x>>y;
        } friend float distance(Point p1, Point p2);
};
float distance(Point p1, Point p2){
        float dx = p2.x - p1.x;
        float dy = p2.y - p1.y;
        return sqrt(dx*dx + dy*dy);
}
```

```
int main(){
        Point p1, p2;
        cout<<"Point 1:"<<endl;
        p1.accept();
        cout<<"Point 2:"<<endl;
        p2.accept();
        float d = distance(p1, p2);
        cout<<"Distance between the 2 points:"
            <<d<<endl;
}
```

13) WAP to create 2 classes BankAcc and Audit. BankAcc holds priv. balance info. Write a friend function in audit that accesses and priv balance info for auditing.

```
#include<iostream>
using namespace std;
class Audit;
class BankAcc{
        int balance;
public:
        BankAcc(int b){
                balance = b;
        }
        friend void AuditAcc(BankAcc& acc, Audit& audit);
};
class Audit{
public: void printAudit(BankAcc& acc){
        AuditAcc(acc, *this);
    }
};
```

```
void AuditAcc (BankAcc & acc, Audit & auditor){
    cout<<" Auditing Acc: "<<endl;
    cout<<"Balance: $ "<< acc.balance <<endl;
}
int main (){
    BankAcc acc1(50000);
    Audit auditor;
    auditor.printAudit (acc1);
}
```

Q1

13/8.

10/9/25

Experiment No-5

To implement the types of constructors.

a) To find sum of nos. between 1 to n using constructor

→

```
#include<iostream>
using namespace std;
class Sum{
    int n;
    int sum;
public:
    Sum(int num){
        n=num;
        sum=0;
        for(int i=1;i<=n;i++){
            sum=sum+i;
        }
    }
    void display(){
        cout<<"Sum of nos:"<<sum<<endl;
    }
};
int main(){
    int n;
    cout<<"Enter value of n:";
    cin>>n;
    Sum s(n);
    s.display();
}
```

```
default:
Sum(){
    n=10;
    sum=0;
    for(int i=1;i<=n;i++)
    sum=sum+i;
}
```

```
copy constructor:
Sum(Sum &obj){
n=obj.n;
sum=obj.sum;
}
```

```
Copy:
Sum s1(5);
Sum s2(s1);
```

```
Default:
Sum s;
s.display();
```

b) Accept, display data for one student.

→
```
#include<iostream>
using namespace std;
class num
class Student{
    string name;
    float perc;
public:
    Student(string n, float p){
    name = n;
    percentage = p;
    }
    void display(){
    cout<<"Name:"<<name<<"\nPercentage:"<<perc;
    }
};
int main(){
    string name;
    float perc;
    cout<<"Enter name:";
    cin>>name;
    cout<<"Enter percentage:";
    cin>>perc;

    Student s1(name, percentage);
    cout<<"Student Details:";
    s1.display();
}
```

Default:
```
Student(){
  name="Parth";
  perc = 98;
}
```

Copy:
```
Student s(Student &s){
name=s.name;
perc=s.perc;
}
```

Default:
```
Student s1();
s1.display();
```

Copy:
```
Student s2(s1);
```

c) Accept data for 2 objs of class and display using constructor:

→
```
#include <iostream>
using namespace std;
class College {
    int roll_no;
    string name, course;
public:
College(int r, string n) {
    roll_no = r;
    name = n;
    course = "Computer Engineering";
}
void display() {
cout<<"\n Roll No: "<<roll_no<<endl;
cout<<" Name: "<<name<<endl;
cout<<" Course: "<<course<<endl;
}
};
int main(){
    College s1(5, "Parth");
    College s2(6, "Aditya");
    s1.display();
    s2.display();
}
```

4) Constructor overloading.

→
```
#include <iostream>
using namespace std;
class college{
  int roll_no;
  string name, course;
  public:
  college(){
        roll_no = 4;
        name = "Parth";
        course = "Computer Engineering";
      }
  College(int r, string n, string c){
      roll_no = r;
      name = n;
      course = c;
    }
  void display(){
  cout<<"Roll No: "<<roll_no<<endl;
  cout<<"Name: "<<name<<endl;
  cout<<"Course: "<<course<<endl;
  }
};
int main(){
    college s;
    college s1(5, "Aditya", "Civil");

    s.display();
    s1.display();
}
```

10/9/25

24/9/25

Experiment No: 6

1) Single Inheritance:

→

```
#include<iostream>
using namespace std;
class Person {
        protected:
        string name;
        int age;
};
class Student : protected Person {
        private:
        int roll;
        public:
        void accept(){
        cout<<" Enter Name: ";
        cin>>name;
        cout<<" Enter Age: ";
        cin>>age;
        cout<<" Enter Roll no: ";
        cin>>roll;
    }
        void display(){
        cout<<" Name and Age: "<<name<<age;
        cout<<" Roll no: "<<roll;
};
  int main(){
        Student S;
        S.accept();
        S.display();
    }
```

2) Multiple Inheritance:

→
```
#include<iostream>
using namespace std;
class Academic{
        protected:
        int marks;
};
class Sports{
        protected:
        int sp_score;
};
class Result:protected Academic, protected Sports{
private:
        float per;
        public:
        void accept(){
        cout<<"Enter marks in academics and sports:";
        cin>>marks>>sp_score;
      }
        void calculate(){
        int total = marks + sp_score;
        per = (total/1200.00)*100;
        cout<<"Result = "<<per<<endl;
    }
};
int main(){
    Result r;
    r.accept();
    r.calculate();
  }
```

3) Multilevel Inheritance:-

→
```
#include<iostream>
using namespace std;
class Vehicle{
        public:
        string brand, model;
};
class Car: public Vehicle{
        protected:
        string type;
};
class eCar : protected Car{
        private:
        int batteryCap;
        public:
        void accept(){
        cout<<" Enter brand Name: ";
        cin>>brand;
        cout<<" Enter Model";
        cin>>model;
        cout<<" Enter type of car: ";
        cin>>type;
        cout<<" Enter Battery Capacity: ";
        cin>>batterycap;
}
 void display(){
        cout<<" Brand and model: "<<brand<<model;
        cout<<" Type of car: "<<type<<endl;
        cout<<" Battery Capacity: "<<battery cap;
}
};
```

```
int main(){
 Car c;
 c.accept();
 c.display();
}
```

4) Hierarchical Inheritance:

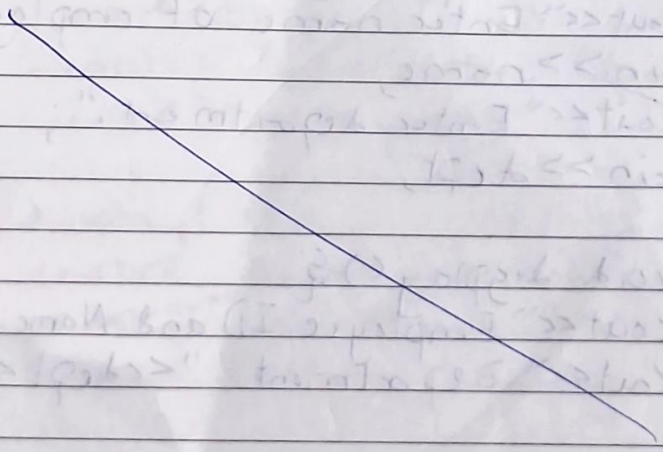
→
```
#include<iostream>
using namespace std;
class Employee{
    protected:
    int emp_id;
    string name;
};
class Manager: public Employee{
    private:
    string dept;
    public:
    void accept(){
    cout<<"Enter Employee ID:";
    cin>>emp_id;
    cout<<"Enter name of employee:";
    cin>>name;
    cout<<"Enter department:";
    cin>>dept;
    }

    void display(){
    cout<<"Employee ID and Name:"<<emp_id<<name;
    cout<<"Department :"<<dept<<endl;
    }
};
```

P.T.O.

```
class Developer: protected Employee{
    private:
    string prog_lang;
    public:
    void acc(){
    cout<<"Enter Programming language:";
    cin>>prog_lang;
}
    void disp(){
    cout<<"Programming Language:"<<prog_lang
    }
};
int main(){
    Manager m;
    m.accept();
    m.display();

    Developer d;
    d.acc();
    d.disp();
}
```



5) Hybrid Inheritance;

→
```
#include <iostream>
using namespace std;
class Person{
protected:
string name;
int age;
public:
void getInfo(){
    cout<<"Enter Name and age:";
    cin>>name>>age;
    }
void disp(){
    cout<<"Name and Age:"<<name<<age<<endl;
    }
};
class Student:public Person{
    protected:
    int roll;
    public:
    void acc(){
    cout<<"Enter roll no:";
    cin>>roll;
    }

    void dispRoll(){
    cout<<"Roll Number:"<<roll<<endl;
    }
};
```

P.T.O.

```
class Academics {
protected:
int m1, m2;
public:
void acceptMarks(){
cout<<"Enter marks for m1 and m2:";
cin>>m1>>m2;
}
 void displayMarks(){
 cout<<"Marks:"<<m1<<m2<<endl;
}
};
class Sports{
    protected:
    int sports_score;
    public:
    void getSports(){
    cout<<"Enter score:";
    cin>>sports_score;
    }
     void showSports(){
     cout<<"sports Marks:"<<sports_score<<endl;
    }
};
class Result: public Student, public Academics, public
Sports {
public:
    void displayResult(){
    int total=m1+m2+sports_score;
    cout<<"Total:"<<total<<endl;
    }
};
```

```
int main(){
    Result r;
    r.getinfo();
    r.disp();
    r.acc();
    r.dispRoll();
    r.acceptMarks();
    r.ddsplayMarks();
    r.getsports();
    r.showsports();
    r.displayResult();
}
```

6) Virtual class:-

```
#include<iostream>
using namespace std;
class CollegeStudent {
protected:
    int st_id, coll_code;
public:
    void accept(){
        cout<<"Enter student id:";
        cin>>st_id;
        cout<<"Enter college id:";
        cin>>coll_code;
    }
    void display(){
        cout<<"Student ID:"<<st_id<<endl;
        cout<<"College code:"<<coll_code<<endl;
    }
};
class Test: virtual public CollegeStudent {
protected:
    float per;
public:
    void acc(){
        cout<<"Enter test percentage:";
        cin>>per;
    }
    void disp(){
        cout<<"Test Percentage:"<<per;
        cin>>per;
    }
};
```

```
class Result : public Test, public Sports {
public:
void getdata() {
        get accept();
        acc();
}

class Sports : virtual public CollegeStudent {
protected:
char grade;
public:
void getsports() {
        cout<< "Enter sports grade: ";
        cin>> grade;
}
void putSports() {
        cout<< "Sports grade: "<< grade << endl;
}
};
class Result : public Test, public Sports {
public:
  void getdata() {
        accept();
        acc();
        getsports();
  }
  void putData() {
        display();
        disp();
        putsports();
  }
};
int main() {
```

```
Result r;
r.getData();
cout<<"\n --- Details - -- \n";
r.putData();
return 0;
}
```

Q-

28/9/23

1/10/25

Experiment No-7

a]

```
#include<iostream>
using namespace std;
class area1{
    public:
    int l, b;
    void area(int a, int b){
    int c = a * b;
    cout<<"Lab Area: "<<c<<" sq.";
    }
    void area (int s) {
    int f = s*s;
    cout<<"Class Area: "<<f<<" sq.";
    }
};
int main(){
    area1 m;
    m.area(20,30);
    cout<<endl;
    m.area(20);
}
```

b]

```
#include<iostream>
using namespace std;
class sum{
public
int i;
void sum(float a[5]){
    float s = 0;
    for(i=0; i<5; i++){
        s+ = a[i];
    }
    cout<<"Sum of 5 float nos:"<<s<<endl;
    }
void sum(int b[10]){
    int s = 0;
    for(i =0; i<10; i++){
        s+ = b[i];
    }
    cout<<"Sum of 10 integer nos:"<< s<<endl;
    }
};
int main(){
    sum s1;
    float [5];
    ind d[10];
    cout<<"Enter 5 float Nos:\n";
    for(int i = 0; i<5; i++){
    cin>> c[i]; }
    cout<<"Enter 10 int nos:\n";
    for(int i = 0; i<10; i++){
    cin>> d[i]; }
    s1.sum(c); s1.sum(d); }
```

c]

```
#include<iostream>
using namespace std;
class num{
    int a;
    public:
    void accept(){
    cout<<"Enter value of a:";
    cin>>a;
    }
    void disp(){
    cout<<"Value of a:"<<a;
    }
    void operator -(){
     a=-a;
    }
};
int main(){
    num n1;
    n1.accept();
    -n1;
    n1.disp();
}
```

11/10/25

d]

```
#include<iostream>
using namespace std;
class num{
    int a,b, c;
    public:
    void accept(){
    cout<<"Enter val of a:";
    cin>>a>>b;
    }
    void disp(){
    cout<<"value of a:"<<a;
    }
    void operator ++(){
    a = ++a;
    }
};
int main(){
    num n1;
    n1.accept();
    ++n1;
    n1.disp();
}
```

28/10/25

Experiment No. 8

①
```
#include <iostream>
#include <string>
using namespace std;
class abc{
public:
string str;
void acc(){
        cout<<"Enter string:";
        cin>>str;
}
abc operator +(abc s){
abc temp;
temp.str = str + s.str;
return temp;
}
void disp(){
    cout<<"Concatenated string:"<<str<<endl;
};
int main(){
abc s1, s2, r;
s1.acc();
s2.acc();
r = s1+s2;
r.disp();
return 0;
}
```

* O/P:

```
Enter string: Hello
Enter string: World
Concatenate string: HelloWorld
```

② #include<iostream>
using namespace std;
class ILogin {
protected:
string name, password;
public:
void accept() {
cout << "Name: ";
cin >> name;
cout << "Password: ";
cin >> password;
}
};
class EmailLogin : virtual public ILogin {
public:
void showEmail() {
cout << "Name " << " " << password << endl;
}
};
class MembershipLogin : virtual public ILogin {
public:
void showMembership() {
cout << "name " << " " << password << endl;
}
};
class Employee : public EmailLogin, public MemberLogin {
public:
void input() {
accept();
}
void display() {
showEmail();

```
    showMembership();
{
};
int main(){
    Employee e;
    e.input();
    e.display();
    return 0;
}
```

28/10/25

Experiment No. 9

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <string>
using namespace std;
int main(){
    ifstream fin;
    ofstream fout;

    fin.open("source.txt");
    fout.open("destination.txt");

    if(!fin){
      cout << "error takes place" << endl;
      return 1;
    }
    char ch;
    while(fin.get(ch)){
        fout.put(ch);
    }
    fin.close();
    fout.close();
    fin.open("source.txt");
    string word;
    int wordcount = 0;
    while(fin >> word){
        wordcount++;
    }
    cout << "The wordcount is:" << wordcount << endl;
```

```
fin.close();
fin.open("source.txt");
String target;
int count = 0;
cout<<"Enter the target"<<endl;
cin>>target;
while(fin>word){
    if(word==target){
        count++;
    }
}
cout<<"The target word found whose occurence is"<<
    count<<endl;
fin.open("source.txt");
int digitcount=0;
int spacecount=0;

while(fin.get(ch)){
    if(isdigit(ch)){
        digitcount++;
    }
    if(isspace(ch)){
        spacecount++;
    }
}
cout<<"Digit count is:"<<digitcount;
cout<<"The space count is:"<<spacecount;
fin.close();
}
```

12/11

5/11/25

Experiment No: 10

a]

```
#include <iostream>
using namespace std;
template <class T>
T sum (T a[], int n) {
    T sum = 0;
for(int i = 0; i<n; i++){
    sum += a[i];
    }
    return sum;
}
int main() {
    int n = 5;
    int inta[n];
    float floata[n];
    double doublea[n];

    cout << "Enter 5 integer numbers: ";
    for(int i=0; i<n; i++){
        cin >> inta[i];
    }
    cout << "Enter 5 integer elements: ";
    for(int i = 0; i<n; i++){
        cin>> floata[i];
      }
    cout << "Enter 5 double numbers: ";
    for(int i = 0; i<n; i++){
        cin >> doublea[i];
}
```

```
cout<<"\nSum of integer numbers="<<sum(int a,n);
cout<<"\nSum of float numbers="<<sum(float a,n);
cout<<"\nSum of double elements="<<sum(double a,n);

}
```

b]

```
#include<iostream>
#include<string>
using namespace std;
template<class T>
T square(T x){
    return x*x;
}
template<>
string square<string>(string s){
        return s+s;
}
int main(){
    int num;
    string str;

    cout<<"Enter an integer:";
    cin>>num;

    cout<<"Enter a string:";
    cin>>str;

cout<<"\n Square of integer "<<num<<"="<<square(num);
cout<<"\n Square of string "<<str<<"="<<square(str);

}
```

c)
```
#include<iostream>
#include<cmath>
using namespace std;
template<class T>
class calc{
    public:
    T a, b;
    void accept(){
    cout<<"Enter 2 nos:";
    cin>>a>>b;
    }
    void add(){
    cout<<"Addition is:"<<a+b<<endl;
    }
    void sub(){
    cout<<"Subtraction is:"<<a-b<<endl;
    }
    void mul(){
    cout<<"Multiplication is:"<<a*b<<endl;
    }
    void div(){
    cout<<"Division is:"<<a/b<<endl;
    }
};
int main(){
cal<double> n;
int ch;
n.accept();
while(1){
cout<<"1.Addition\n2.Subtraction\n3.Multiplication\n
    4.Division\n5.Exit";
cout<<"Enter choice:";
```

```
cin >> ch;
switch (ch) {
        case 1: n.add(); break;
        case 2: n.sub(); break;
        case 3: n.mul(); break;
        case 4: n.div(); break;
        case 5: exit(0);
        default: "Wrong choice:";
        break;
    }
}
```

5/11/25

Experiment No. 11

i)

```
#include <iostream>
#include <vector>
#include<ctype>
using namespace std;
int main(){
    vector<int> vec[5);
    int i;
    cout<<"Enter 5 vector elements:";
    for(i=0; i<5; i++){
        cout<<vec[i]<<endl;
    }
    cout<<" Modified elements:";
    for(i=0; i<5; i++){
    vec[i] = vec[i] +i*2;
    }
    for(i=0; i<5; i++){
        cout<<vec[i]<<" ";
    }
    cout<<endl;
    int scalar;
    cout<<"Enter a scalar val to multiply:";
    cin>>scalar;
    cout<<" After multiplying:";
    for(i=0; i<5; i++){
    vec[i] = vec[i] * scalar;

    for(i=0;i<5;i++){
    cout<<vec[i]<<" "; }
    cout<<endl;
}
```

Experiment no.11 (using iterators)

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

int main(){
    vector<int>vec(5);
    int scalar;
    cout<<"Enter 5 vector elements:";
    vector<int>::iterator it = vec.begin();
    while(it != vec.end()){
        cin>>*it;
        ++it;
    }
    cout<<"Modified Elements:";
    it = vec.begin();
    while(it != vec.end()){
        *it = *it * 2;
        ++it;
    }
    it = vec.begin();
    while(it != vec.end()){cout<<*it<<" ";
    ++it;
    cout<<"Enter a scalar to multiply:";
    cin>>scalar;
    cout<<"After multiplying:";
    it = vec.begin() while(it != vec.end()){
    *it = *it * scalar; ++it; }
    it = vec.begin();
    while(it != vec.end()){ cout<<*it<<" ";
    ++it;}cout<<endl;
}
```

12/11

5/11/25

Experiment No. 12

1)
```
#include<iostream>
#include<stack>
#include<ctype>
using namespace std;
int main(){
stack<int> v;
v.push(1);
v.push(2);
v.push(3);
v.push(4);
v.push(5);

if (v.empty()){
    cout<<"stack is empty";
}
else{
    cout<<"\n stack is not empty";
}

cout<<"\n size: "<<v.size();
cout<<"\n topmost: "<<v.top();

cout<<"\n stack:";
while(!v.empty())
{cout<<v.top()<<" "; v.pop;
}
cout<<"\n size after popping :"<<v.size();
}
```

2)
```
#include<iostream>
#include <queue>
#include <ctype>
using namespace std;
int main(){
    queue <int> v;
    v.push (11);
    v.push (22);
    v.push (33);
    v.push (44);
    v.push (55);
    if(v.empty()){
    cout<<"\n queue empty";
}
    else{
    cout<<"\n queue is not empty";
}
cout<<"\n Size: "<< v.size();
cout<<"\n. Front: "<< v. front();
cout<<"\n. Back: "<<v. back();
cout<<"\n Queue:";
while (!v. empty()){
        cout<<" v.front()<<" "; } v.pop();
}
cout<<"\n Size after popping:"<<v.size();
}
```

Qn
12/11