

Experiment : 1

- 1] Write a program to declare a class Student having data members as roll number, name accept and display data for one object

```
#include <iostream>
using namespace std;
class student
{
    int roll_no;
    string name;
public:
    void accept ()
    {
        cout << "Enter roll number: "
        cin >> roll_no;
        cout << "Enter name: ";
        cin >> name;
    }
    void display ()
    {
        cout << "\n student, Roll No: " << roll_no.
        cout << "\n Name: " << name << endl;
    }
};

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

Output :

Student Roll No : 27

Name : Adwait

- 2] Write a program to declare a class group having data members as : book name, price and no of pages Accept this 'data' for 2 object and display name of book having greater price.

```
#include <iostream>
```

```
using namespace std;
```

```
class BookBook
```

```
{
```

```
    string book_name;
```

```
    float price;
```

```
    int page;
```

```
public:
```

```
void accept()
```

```
{
```

```
    cout << "Enter book name: ";
```

```
    cin >> book_name;
```

```
    cout << "Enter price: ";
```

```
    cin >> price;
```

```
    cout << "Enter number of pages: ";
```

```
    cin >> pages;
```

```
}
```

```
float get_price()
```

```
{
```

```
    return price;
```

```
}
```

```
string getName()
{
    return book-name;
}
int main()
{
    Book b1, b2;
    b1.accept();
    b2.accept();
    if (b1.get price() > b2.get price())
        cout << "Book with greater price : "
             << b1.getName() << endl;
    else
        cout << "Book with greater price : "
             << b2.getName() << endl;
    return 0;
}
```

Output

Enter book name: Cpp Basics

Enter price: 300.

Enter number of pages: 250

Enter book name: Java

Enter price: 450.

Enter number of pages: 300

Book with greater price: Java.

3] Write a program to declare a class time accept the time in HH:MM:SS and convert it into total second and display

```
#include <iostream>
using namespace std
class Time
{
    int hours, minutes, seconds
public:
    void accept()
    {
        cout << "Enter time in HH:MM:SS format : ";
        cin >> hours >> minutes >> seconds;
    }
    void convert_and_display()
    {
        int total_seconds = hours * 3600 + minutes * 60 + seconds;
        cout << "Total time in seconds : " << total_seconds;
    }
};

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

Output

Enter time in HH:MM:SS format : 1:30:45
Total time in second : 5495

Q
30/7/25

Experiment 12

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

```
#include <iostream>
```

```
using namespace std;
```

```
class city {
```

```
    string name;
```

```
    int population;
```

```
public:
```

```
    void accept();
```

```
}
```

```
cout << "Enter city name:";
```

```
cin >> name;
```

```
cout << "Enter population:";
```

```
cin >> population;
```

```
}
```

```
void display();
```

```
}
```

```
cout << "city :" << name;
```

```
cout << "population :" << population;
```

```
}
```

```
;
```

```
int main()
```

```
{
```

```
    city c[5];
```

```
    for (int i=0; i<5; i++)
```

```

cout << "Enter details for city" << i + 1;
c[i].accept();
int max = 0;
for (int i = 1; i < 5; i++)
{
    if (c[i].population > cities[c[max]].population)
        max = i;
}
cout << "city with highest population : ";
c[max].display();
return 0;
}

```

Output

```

Enter details of city : 1
Enter name of city : pune
Enter the population of city : 10000
Enter the details of city : 2
Enter the name of city : Mumbai
Enter the population of city : 1750000
Enter the details of city : 3
Enter name of the city : NYC
Enter the population of city : 17772
Enter the details of city : 4
Enter name of the city : los angle
Enter the population of city : 90051
Enter the details of city : 5
Enter name of the city : Tokyo
Enter population of the city : 7000

```

City with highest population is Mumbai.

2. Write a program to access a class Account having account number, account name and balance. Input data is to account and the output of the program is to show balance if balance is less than 5000 and display them.

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

```
class account
```

```
{public:
```

```
int acc_no;
```

```
string name;
```

```
void credit();
```

```
}
```

```
cout << "Enter account number:";
```

```
cin >> acc_no;
```

```
cout << "Enter balance:";
```

```
cin >> balance;
```

```
void interest()
```

```
{
```

```
if (balance >= 5000)
```

```
balance = balance * 0.10;
```

```
}
```

```
void display()
```

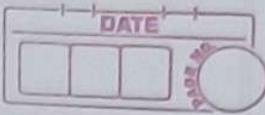
```
cout << "Print No. " << acc_no
```

```
cout << "Balance " << balance;
```

```
};
```

```
int main()
```

```
{
```

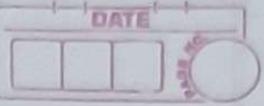


```
account acc [10]
for (int i=0; i<10; i++)
{
    cout << "Account " << i + 1 << endl;
    acc[i].accept ();
}
for (int i=0; i<10; i++)
{
    acc[i].interest ();
}
cout << "Account details after applying interest";
for (int i=0; i<10; i++)
{
    acc[i].display ();
}
return 0;
```

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

```
#include <iostream>
#include <string.h>
using namespace std;
class staff {
public
    char name[50], post[50];
    void accept()
    {
        cout << "Enter name and post : ";
        cin >> name >> post;
    }
    void display()
    {
        cout << "HOD are :" << name << endl;
    }
};

int main()
{
    staff s[5];
    for (int i=0; i<5; i++)
    {
        s[i].accept();
    }
    for (int i=0; i<5; i++)
    {
        s[i].display();
    }
}
```



8

```
if (str cmp (S[i].post, "HOD") == 0)  
    S[i].display();
```

Output :

Staff 1 name : John - Snow
Staff 1 post : HOD

Staff 2 name : Walter - White
Staff 2 post : Janitor

Staff 3 name : Bob
Staff 3 post : Assistant

Staff 4 name : David
Staff 4 post : Professor

Staff 5 name : Sean
Staff 5 post : Lecturer

Staff members who are HOD:
John - Snow.

Qn
30/7/25

2] Output :

Enter account number : 201

Enter balance : 4000

Enter account number : 202

Enter balance : 5000

Enter account number : 203

Enter balance : 2500

Enter account number : 204

Enter balance : 6000

Enter account number : 205

Enter balance : 7000

Enter account number : 206

Enter balance : 3000

Enter account number : 207

Enter balance : 8000

Enter account number : 208

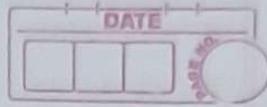
Enter balance : 1000

Enter account number : 209

Enter balance : 12000

Enter account number : 210

Enter balance : 19000



Account details after applying interest

Account no: 201

Balance: 8900

Account no: 202

Balance: 5500

Account no: 203

Balance: 2750

Account no: 204

Balance: 6600

Account no: 205

Balance: 7700

Account no: 206

Balance: 3300

Account no: 207

Balance: 8800

Account no: 208

Balance: 1100

Account no: 209

Balance: 13200

Account no: 210

Balance: 15400

On

30/7/25

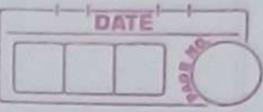
Experiment : 3.

- 1] W.A.P 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 the object.

```
#include <iostream>
using namespace std;
class Book
{
    string book_title;
    string author_name;
    int price;

public:
    void accept()
    {
        cout << "Enter the book title:"; cin >> book_title;
        cout << "Enter the author name:"; cin >> author_name;
        cout << "Enter the price of the book:"; cin >> price;
    }

    void display()
    {
        cout << "Book Title :" << book_title << endl;
        cout << "Author Name :" << author_name << endl;
        cout << "Author Name :" << author_name << endl;
        cout << "Price :" << price << endl;
    }
}
```



int main ()

{

Book b;

Book * p = & b;

p → accept();

p → display();

return 0;

}

Output .

:>>> " enter book name " >> two

:>>> " enter book type " >> one

:>>> " enter book price " >> two

:>>> " enter book quantity " >> two

() wildeib . two

() tforia <- eight

" enter book <- eight >> " -> eight book . eight " >> two

" need <- eight >> " -> eight book quantity . eight " >> two

() nimm . five

(2) tribute

(3) wildeib . two

b] WAP to declare a class 'student' having data members roll-no and percentage. Using this pointer invoke member function to accept and display this data for one object of a class.

```
#include <iostream>
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 << "the roll no is -" << this->roll-no;
        cout << "the percentage is :-" << this->perc;
    }
};

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

2] WAP to demonstrate the use of nested class

```
#include <iostream>
using namespace std;
class demo {
public:
    class demo2 {
public:
        int roll no;
        char name[50];
        void accept () {
            cout << "enter roll no:- ";
            cin >> roll no;
            cout << "enter the name :- ";
            cin >> name;
        }
        void display () {
            cout << "the roll no is:- " << roll no << endl;
            cout << "the name is:- " << name << endl;
        }
    };
    demo2 d1;
    student.accept ();
    student.display ();
}
```

Experiment : 4.

```
① #include <iostream>
using namespace std;

class result2;
class result1;

int m1;
public:
void accept()
{
cout << "Enter marks of first student : ";
cin >> m1;
}

friend void calculate(result1 p, result2 q);
};

class result2;
int m2;
public:
void accept2()
{
cout << "Enter marks of second student : ";
cin >> m2;
}

friend void calculate(result1 p, result2 q);
};

void calculate(result1 p, result2 q)
{
int total = (p.m1 + q.m2) / 2;
cout << "Total marks of both students : " << total;
}
```



```
int main ()  
{  
    result, k;  
    result2 b;  
    k.accept();  
    b.accept2();  
    calculate (k, b);  
}
```

- 2] Swapping 2 nos using without friend function object as function argument.

```
#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;  
        cout << "after swapping :- " << "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.display ();  
}
```

3] 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 {
public:
    int num1;
    void accept1() {
        cout << "enter first number" << endl;
        cin >> num1;
    }
    friend void greatest_num (A P, B q);
};

class B {
public:
    int num2;
    void accept2() {
        cout << "enter second number" << endl;
        cin >> num2;
    }
    friend void greatest_num (A P, B q);
};
```

```
void greatest_num (A P, B q)
```

{

```
if (P.num1 > q.num2)
```

{

```
cout << "greatest number:-" << p.num1 << endl;
```

{

```
else
```

{

```
cout << "greatest number:-" << q.num2 << endl;
```

{

{

```
int main ()
```

{

```
A k;
```

```
B f;
```

```
k.accept1();
```

```
f.accept2();
```

```
greatest_num(k, f);
```

{

Q] Swapping 2 number from same class by using concept of friend function.

```
#include <iostream>
using namespace std;
class demo
{
    int a, b;
public:
    void accept()
    {
        cout << "enter 2 numbers : - ";
        cin >> a >> b;
    }
    void display()
    {
        cout << "value of a : - " << a;
        cout << "value of b : - " << b;
    }
    friend void swap num S.(demo &t);
};

void swap num (demo &t)
{
    int temp = t.a;
    t.a = t.b;
    t.b = temp;
}

int main()
{}
```

demo k;
k.accept();
swap_nums(k);
k.display();
}

5] WAP to swap two numbers of different class
using concept of friend function

```
#include <iostream>
using namespace std;
class B;
class A
{
    int a;
public:
    void accept()
    {
        cout << "enter a:-" << endl;
        cin >> a;
    }
    void display()
    {
        cout << "Value of a:-" << a;
    }
    friend void swap_numbers(A &P, B &Q);
};

class B
{
    int b;
public:
```

```

void accept_1()
{
    cout << "enter a:- " << endl;
    cin >> a;
}

void display_1()
{
    cout << "value of a:- " << a;
}

friend void swap_numbers(A&P,B&Q);
void swap_numbers(A&P,B&Q)
{
    int temp = P, Q;
    P = Q, Q = temp;
    Q = temp;
}

int main()
{
    A();
    B();
    K().accept_1();
    K().accept_2();
    swap_numbers(K, B);
    K().display_1();
    B().display_2();
}

```

Qn
14/8

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

```
#include <iostream>
using namespace std;
class B;
class A;
{
    int a;
public:
    void accept()
    {
        cout << "Enter the first number:" << endl;
        cin >> a;
    }
    friend sum(Ap, Bf);
};

class B
{
    int b;
public:
    void accept()
    {
        cout << "Enter the second number:" << endl;
        cin >> b;
    }
    friend int sum(Ap, Bf);
};
```

int sum (A p, B q)

{

int sum;

sum = p a + q b;

return sum;

}

int main ()

{

A k;

B b;

k. accept ();

b. accept ();

cout << "The sum of 2 numbers is:" << sum (k,

b);

(A & A) odd man out bias brief

(1) wolf's bias

"a >" : a to out " > true

"b >" : b to out " > true

(A & A) odd man out bias brief

(A & A) odd man out bias

part = don't true

! d. t = a. t

! don't = ! a. t

7] WAP with a class number that contains a private integer. Use a friend function swapnumber (number s, number s) to swap the private value of two number objects.

```
#include <iostream>
using namespace std;
class A
{
    int a, b;
public:
    void accept()
    {
        cout << "Enter two number : " << endl;
        cin >> a >> b;
    }
    friend void swapnumber (A &B);
    void display()
    {
        cout << "Value of a : " << a;
        cout << "Value of b : " << b;
    }
    friend void swapnumber (A &B);
};
void swapnumber (A &B)
{
    int temp = t.a;
    t.a = t.b;
    t.b = temp;
}
```

int main ()

{

A k;

k.accept();

swap numbers (k);

k.display();

}

8) Define two classes Box and cube, each having a private volume. Write a friend function find greater (Box, cube) that determine which object has a larger volume.

```
#include <iostream>
using namespace std;
class cube;
class box;
int l, b, h, V1;
public:
friend accept();
```

cout << "Enter the dimensions of the box";

cin >> l >> b >> h;

}

friend void greater volume (box k, cube q);

}

class cube

{
int side, V2;

public:

void accept ()

{

cout << "Enter the dimensions of the cube";
cin >> side;

}

friend void greater volume (box p, cube q);

{

void greater volume (box p, cube q)

{

p.V1 = p.l.p.b;

q.V2 = q.side * q.side * q.side;

if (p.V1 > q.V2)

{

cout << "The box having greater volume is:";
<< p.V1 << endl;

}

else .

{

cout << "The box having greater volume is:";
<< q.V2 << endl;

}

{

```
int main()
{
    box k;
    cube b;
    k.accept();
    b.accept();
    greater volume(k, b);
}
```

- 9) 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 cmplx
{
    int re, im;
public:
    void accept()
    {
        cout << "Enter the real and imaginary parts";
        cin >> re >> im;
    }
    friend cmplx add cmplx(cmplx x1, cmplx x2)
    {
        cmplx result;
        result.re = x1.re + x2.re;
        result.im = x1.im + x2.im;
        return result;
    }
    void display()
    {
        cout << re << "+" << im << "i";
    }
}
```

friend complex <operator>+(complex <operator>C1, complex <operator>C2,
complex <operator>result);

{

complex <operator>+complex <operator>(complex C1, complex C2,
complex result);

result.re = C1.re + C2.re;

result.im = C1.im + C2.im;

return result;

}

int main()

{

complex k, b, r;

k.accept();

b.accept();

r = add complex(k, b, r);

r.display();

}

Q. Create a class student with private data members: name and three subject marks. write a friend function calculate average (student) that calculate and display the average marks.

```
#include <iostream>
using namespace std;
class student
{
    char name [50];
    int m1, m2, m3;
public:
    void accept ()
    {
        cout << "Enter the name of the student";
        cin >> name;
        cout << "Enter m1:";
        cin >> m1;
        cout << "Enter m2:";
        cin >> m2;
        cout << "Enter m3:";
        cin >> m3;
    }
    friend void calculate (student p);
    void calculate avg (student p)
    {
        int avg;
        avg = (p.m1 + p.m2 + p.m3) / 3;
        cout << "The average of the student marks is:" << avg;
    }
}
```

```

int main()
{
    student k;
    k.accept();
    calculate_avg(k);
}

```

- 1] Create three classes, Alpha, Beta, 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
{
    int a;
public:
    void accept()
    {
        cout << "Enter a:";
        cin >> a;
    }
    friend void sum(Alpha p, Beta q, Gamma x);
};

class Beta
{
    int b;
public:
    void accept()
    {
        cout << "Enter b:";
        cin >> b;
    }
    friend void sum(Alpha p, Beta q, Gamma x);
};

class Gamma
{
    int c;
public:
    void accept()
    {
        cout << "Enter c:";
        cin >> c;
    }
    friend void sum(Alpha p, Beta q, Gamma x);
};

void sum(Alpha p, Beta q, Gamma x)
{
    cout << "Sum is " << p.a + q.b + x.c;
}

```



```
void accept 2()
```

```
{
```

```
cout << "Enter b :";
```

```
cin >> b;
```

```
}
```

```
friend void sum (alpha p, beta q, gamma x);
```

```
{
```

```
class gamma :
```

```
{
```

```
int c;
```

```
public :
```

```
void accept 3()
```

```
{
```

```
cout << "enter c :";
```

```
cin >> c;
```

```
}
```

```
friend void sum (alpha p, beta q, gamma x);
```

```
{
```

```
void sum (alpha p, beta q, gamma x)
```

```
{
```

```
int sum ;
```

~~sum = p.a + q.b + x.c;~~~~cout << "The sum is : " << sum << endl;~~~~{~~

```
int main ()
```

```
{
```

```
alpha k;
```

```
gamma z;
```

```
beta s;
```

```
k.accept 1();
```

```
q.accept 2();
```

```
z.accept 3();
```

```
sum (k, q, z);
```

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

```
#include <iostream>
using namespace std;
class point
{
private:
    int x, y;
public:
    void accept()
    {
        cout << "Enter x and y : ";
        cin >> x >> y;
    }
    friend void find_difference(point p1, point p2);
    void find_difference(point p1, point p2)
    {
        int diff_x = p2.x - p1.x;
        int diff_y = p2.y - p1.y;
        cout << "Difference in x : " << diff_x << endl;
        cout << "Difference in y : " << diff_y << endl;
    }
    int main()
    {
        point a, b;
        cout << "Enter point 1 : " << endl;
        a.accept();
```

```

cout << "Enter point 2 :" << endl;
b. accept ();
Find difference (a, b);
return 0;
}

```

- 8] Create two classes : Bank account and Audit.
 Bank Account hold 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 {
public:
void accept ();
{
cout << "Enter the .balance " << endl;
cin >> balance;
}
friend void access balance (bank account k);
void access balance (bank account k)
{
int put balance;
put balance = k.balance;
cout << "the balance for auditing is "
<< put balance;
}

```



int main ()
{

bank account x;

x.accept();

access balance (x);

3.

On

26/8

< project > obuleni #

: hts eredezeket pozit

: fibo zenek

: taurin dual zenek

: orszakot tri

: sikhud

: () teleni bori

: "there >>" exaktol. vdt sehol? >> tungs

: exaktol << vds

: (k. taurin dual) exaktol exaktol bori

: (k. taurin dual) exaktol exaktol bori

: exaktol tud tri

: exaktol, k. exaktol tud

: exaktol vdt >> tungs

: exaktol tud >>



Experiment : 5

a) WAP to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

```
#include <iostream> sum (int n)
using namespace std;
class sum {
private:
    int num, summy = 0;
public:
    sum () {
        num = 5
        for (int i = 0; i < num; i++)
            sum += i;
        summy = sum + num;
    }
    cout << "Sum of n numbers is " << summy;
    sum (sum and t) {
        num = t, num;
        for (int i = 0; i < num; i++)
            summy += i;
        summy = summy + num;
    }
    cout << "Sum of n numbers is " << summy;
};
```

int main ()

{ int sum1 = 0;

int sum2 = 0;

int sum3 = 0;

int sum4 = 0;

}

int main ()

{ int sum1 = 0;

int sum2 = 0;

int sum3 = 0;

int sum4 = 0;

return 0;

< program > submitted by

: Date : 2023-08-10

- b) 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>
```

```
#include <string>
```

```
using namespace std;
```

```
class student {
```

```
private:
```

```
string name;
```

```
float perc;
```

```
public:
```

```
student ()
```

```
cout << "name of student";
```

```
cin >> name;
```

```
cout << "percentage of student"
```

```
cin >> perc;
```

```
void display ()
```

```
cout << "name of student is " << name;
```

```
cout << "Percentage of student " << perc;
```

student (student & s)

name = s.name;
perc = s.perc;

cout << "name of student is "; << name;
cout << "percentage of student is "; << perc;

int main ()

student k;

k.display();

student s ("Adwait", 84);

student z(5);

student (string n, float p) = an - User
name = n;
perc = p;

an - User > "tribute for exam" > true

an - User > "tribute for an user" > true

an - User > "tribute for exam" > true

() is an tri

("Adwait", 84) is an tri

c) Define a class "college" members variable as roll-no, name, course. WAP using constructors with default value as "Computer Engineering" for course. Accept this data for two object of class and display the data.

```

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

class college
{
    int roll-no;
    string name;
    string course;
public:
    college (int r, string s, string c = "CSE")
    {
        roll-no = r;
        name = s;
        course = c;
    }
    cout << "name of student " << name;
    cout << "roll no of student " << roll-no;
    cout << "course of student " << course;
};

int main ()
{
    college .k(24, "Adwait");
    college .l(25, "Devang", "Civil");
}
  
```

d) WAP to demonstrate constructor overloading.

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

```
class student {
    private:
```

```
    string name;
```

```
    float perc;
```

```
public:
```

```
student ()
```

```
name = "Adwait"
```

```
perc = 98.6;
```

```
void display ()
```

```
cout << name;
```

```
cout << perc;
```

```
student (string n, float p)
```

```
name = n;
```

```
perc = p;
```

```
cout << "name of student is " << name;
```

```
cout << "percentage of student ." << perc;
```

```
student (student ss)
```

student (student & s)

```

name = S.name;
perc = S.perc;
cout << "name is " << name;
cout << "percentage is " << perc;
  
```

int main ()

student K;

K.display();

student r ("Adwait", 68, 28);

student S(r);

}

: stored

: error write

: need. read

: valid

: attribute

: invalid

: error

: d, 8P = need

: S

: () valid

: error >> true

: need >> true

: (d, t, a, l, n, g, r, e, t, u, r, e) tribute

: (n = error)

: (d = need)

: (error >> "d: tribute for error" >> true)

: (need >> "tribute for registration" >> true)

: (2 & tribute) tribute



Experiment : 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 the name : ";
cin >> name;
cout << "Enter the age : ";
cin >> age;
cout << "Enter the roll number of student : ";
cin >> roll;
```

```
void display
```

```
{
```

```
cout << "Name : " << name;
```

```
cout << "Age : " << age;
```

```
cout << "Roll number : " << roll;
```

```
};
```

```
};
```

```
int main ()
```

{

```
student S1;
```

```
S1.accept();
```

```
S1.display();
```

}

2] Multiple Inheritance

```
#include <iostream>
```

```
using namespace std;
```

```
class Academic
```

}

Protected:

```
int marks;
```

}

```
class Sports
```

}

protected:

```
int S-score;
```

}

```
class Result : Protected Academic, Sports
```

}

private:

```
int total;
```

public:

```
void accept()
```

}

```
cout << "Enter academic marks and sports score"
```

```
cin >> marks >> S-score;
```

}



```
void calculate ()
```

$$\text{total} = (\text{marks} + \text{s. score}) / 200 \times 100$$

```
}
```

```
void display ()
```

```
{
```

```
cout << "Marks " << Marks;
```

```
cout << "Sports Score " << S. score;
```

```
cout << "Total Score " << total;
```

```
}
```

```
};
```

```
int main ()
```

```
{
```

```
Result R;
```

```
R. accept ();
```

```
R. calculate ();
```

```
R. display ();
```

```
}
```

3] Multi-level Inheritance

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

```
class Vehicle
```

```
{protected:
```

```
string brand;
```

```
string model;
```

```
} class car : protected vehicle
```

```
{protected:
```

```
string type;
```

```
} class EV : protected car
```

```
{private:
```

```
int battery;
```

```
public:
```

```
void accept()
```

```
{cout << "Enter brand model type and battery";
cin >> brand >> model >> type >> battery;
}
```

~~```
void display()
```~~

```
cout << "Brand name " << brand;
```

```
cout << "Model name " << model;
```

```
cout << "Type :" << type;
```

```
cout << "Battery capacity " << battery << "RWH"
```

```
}
```

int main()  
{  
 Employee e;  
 e.accept();  
 e.display();  
}

## Q] Hierarchical Inheritance

```
#include <iostream>
using namespace std;
class Emp
{
```

protected:

string name;

int emp\_id;

};

class Manager : Protected\_Emp

private

string dept;

public :

void accept();

cout << "Enter name, emp\_id and dept";

cin >> name >> emp\_id >> dept;

};

void display()

cout << "Name " << name;

cout << "Emp\_id " << emp\_id;

cout << "Dept " << dept;

};

```
class Developer : Protected Emp
{
private:
 string language;
public:
 void input()
{
 cout << "Enter Name, emp-id & programming
language";
 cin >> name >> emp-id >> language;
 }
 void output()
{
 cout << "Name " << name;
 cout << "Emp-id " << emp-id;
 cout << "Dept " << dept;
 }
};
int main()
{
 Manager m;
 m.accept();
 m.display();
 Developer d;
 d.input();
 d.output();
 return 0;
}
```

5) Hybrid :

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

```
class Person
```

```
{
```

```
protected:
```

```
string name;
```

```
int age;
```

```
}
```

```
class student : protected person.
```

```
{
```

```
protected:
```

```
int roll;
```

```
Academics aca;
```

```
}
```

```
class Academies
```

```
{
```

```
public:
```

```
int marks;
```

```
}
```

```
class sports
```

```
{
```

```
public:
```

```
int s_score;
```

```
}
```

```
class Result : public sports, public student
```

```
{
```

```
private
```

```
int total;
```

```
};
```

```
void accept ()
{
```

```
 cout << "Enter name and age";
 cin >> name >> age;
 cout << "Enter Roll";
 cin >> roll;
 cout << "Enter academic and sports marks";
 cin >> aca_marks >> s_score;
}
```

```
void calculate ()
```

```
{
 total = (aca_marks + s_score) / 2;
}
```

```
void display ()
```

```
{
 cout << "Name" << name;
 cout << "age" << age;
 cout << "Roll" << roll;
 cout << "Aca. marks" << aca_marks;
 cout << "Sports score" << s_score;
 cout << "Total" << total;
}
```

```
int main ()
```

```
{
 Result r;
```

```
 r. accept ();
 r. calculate ();
 r. display ();
}
```

## 6] Virtual base class

```

#include <iostream>
using namespace std;
class cyl - student
{
protected:
 string cyl - code;
};

class Test : virtual public cyl - student
{
protected:
 int perc;
};

class sports : virtual public cyl - student
{
protected:
 int s - score;
};

class Result : Protected test, sports
{
public:
 void accept()
 {
 cout << "Enter cyl code, perc, S - score";
 cin >> cyl code >> perc >> S - score;
 }
};

```

void display ()

{  
cout << "city code = " << city code;  
cout << "percentage " << per;  
cout << "sports score " << S - score;  
}

int main ()  
{

Result r;  
r. accept ();  
r. display ();

Qn  
26/9/25

## Experiment : 7

a) Write a program using function overloading to calculate the area of a laboratory and area of each classroom

```
#include <iostream>
using namespace std;
class area
{
public
 int l, b;
 void area (int a, int b)
 {
 int c = a * b;
 cout << "laboratory Area : " << c;
 }
 void area (int s)
 {
 int b = s * s;
 cout << "class Area " << b;
 }
};

int main ()
{
 area m;
 m.area (20, 30);
 m.area (20);
}
```

"area of rectangle" > true

"area of square" > true

Q) WAP using function overloading to calculate  
the sum of 5 float value and sum of 10 int value

```
#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 10 integers numbers ";
 }
 int main()
 {
 sum s1;
 float c[5];
 int d[10];
 cout << "Enter 5 float Nos: ";
 for(int i=0; i<5; i++)
 cin >> c[i];
 cout << "Enter 10 integer nos: ";
 }
}
```



S<sub>1</sub>. sum(c);  
S<sub>1</sub>. sum(d);  
}

( ) marks tri

3  
i/p menu

: O : fderron

- c) WAP to implement binary operator when used with the object so that the numeric data members of the class is negated.

```
#include <iostream> mld mi ut qAW [b
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 n;

cin >> n;

cout << n;

}

- Q) WAP to implement the unary + operator associated with the object so that the numeric data member of the class is incremented.

#include <iostream>

using namespace std;

class num

{

int a,b,c;

public:

void accept()

cout <"Enter value of a";

cin >> a >> b >> c;

void disp()

cout <"Value of a:" << a;

void operator ++()

a = a + 1;

8: truncated

int main ()

{ int n; cout <> " + " <> n; cout <> " is a number at TADL is "

num.n; } & heterotaxon et plus unites

n.accept(); } & sfn " this together with

++n;

n disp(); }

Qn  
12/11

< griotte > baubri #  
• Bt. exaderant oulu  
griotte m. leeks

• etc. anisette  
• : rhabd  
(2 griottes) griotte m.

; 2 = etc.

( ) anisette

; " 4 = etc.

(file griotte m.) + entree de la bistro

; etc. jala + etc. = etc.

( ) filet bistro

; etc >> tuc

O vien tri

{ 2, (" rep. q") < 2, (" sup. ") 12 anisette  
; 2 + 2

## Experiment : 8

a] WAP to overload the '+' operator so that two strings can be concatenated. Eg, "xyz" + "pqr" then output will be "xyzpqr".

```
#include <iostream>
using namespace std;
class m_string
{
 string str;
public:
 m_string (string s)
 {
 str = s;
 }
 m_string ()
 {
 str = "";
 }
 void operator + (m_string obj)
 {
 str = str + obj.str;
 }
 void disp ()
 {
 cout << str;
 }
};

int main ()
{
 m_string S1 ("xyz"), S2 ("pqr"), S3;
 S1 + S2;
 cout << "concatenated string : ";
}
```

S<sub>3</sub>=S<sub>1</sub>;  
S<sub>3</sub>.display();  
}

[ii] WAP to create a base class ILogin having data members name and password. Declare accept() function virtual. Derive EmailLogin and MembershipLogin classes from ILogin. Display Email Login details and membership login details of the employee

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

```
class ILogin
```

```
{protected
```

```
string name;
```

```
string pass;
```

```
public:
```

```
virtual void accept()
```

```
{
```

```
cout << "Enter name and password:";
```

```
cin >> name >> pass;
```

```
virtual void disp()
```

```
{cout << "Name" << name << "
```

```
cout << "Password" << pass;
```

```
}
```

```
}
```

class elogin : public login

{  
string email;

string pass;

public:

void accept ()

{

cout << "Enter Email and Password"

cin >> email >> pass;

}

void disp ()

{

cout << "Email " << email;

cout << "Password " << pass;

void ~~del~~

{

class mlogin : public login

{

string mid;

string pass;

public:

void accept ()

{

cout << "Enter M-id and Password"

cin >> mid >> pass;

{

void disp ()

{

cout << "M-id " << mid << " Password " << pass;

{

int main ()

{

ilogin \* iptr

ilogin i;

ilogin e;

mlogin m;

iptr = ? i;

iptr → accept();

iptr → display();

iptr = em;

iptr → accept();

iptr → disp();

return 0; /\* several principles review \*/

}

Qn  
12/11

p: transferred

< inserted > obufrai # li

< inserted > obufrai #

< e > obufrai #

( ) same. itai

' aid inserted di.

' tuad. inserted fa

" aid inserted " rega. tuad

" tuad. inserted " add. aid

" aid! " tuad

((d)).top.aid). elictu

((d)) tuad. tuad

(( )) escl. aid

(( )) escl. tuad

; "filldeesrue benedek elit" > tuad

; (" bat reverse ") rega. aid

; (o = trans. word tri

; (word << aid). elictu

; ++trans. word

; trans. word >"; trans. word;" > tuad

## Experiment : 9

```
1] #include <iostream>
#include <fstream>
#include <string>
int main ()
```

```
if stream fin;
of stream bout;
fout.open ("destination.txt");
fin.open ("source.txt");
if (!fin)
cout << "Error opening source file";
return 1;
```

```
char ch;
while (fin.get (ch))
fout.put (ch);
```

```
fin.close ();
fout.close ();
cout << "File opened successfully";
```

```
fin.open ("source.txt");
```

```
int wordcount = 0;
```

```
string word;
```

```
while (fin >> word)
```

```
wordcount++;
}
```

```
cout << ".Wordcount :" << wordcount;
```

```
bin.close();
bin.open ("source.txt");
string target;
int count = 0;
cout << "Enter target";
cin >> target;
while (bin >> word)
{
 if (word == target)
 count++;
}
cout << "Word occurrence:" << count;
bin.close();
bin.open ("source.txt");
int digitcount = 0;
int spacecount = 0;
while (bin.get (ch))
{
 if (isdigit (ch))
 digitcount++;
 if (isspace (ch))
 spacecount++;
}
cout << "Digit" << digitcount;
cout << "Space" << spacecount;
bin.close();
return 0;
```

Q  
12/11

## Experiment : 10

```

1] #include <iostream>
#include <math>
using namespace std;
template <class T>
class A
{
 T m1, m2;
public:
 void accept()
 {
 cout << "Enter the first number";
 cin >> m1;
 cout << "Enter the second number";
 cin >> m2;
 }
 void calc()
 {
 int choice;
 cout << "Simple Calculator Menu - \n";
 cout << "1. Addition \n";
 cout << "2. Multiplication \n";
 cout << "3. Division \n";
 cout << "4. Subtraction \n";
 cout << "5. Square root \n";
 cout << "6. Percentage (m1 is what % of m2) \n";
 cout << "7. Power (square of both numbers) \n";
 cout << "8. Trigonometric (sin values) \n";
 cout << "Enter your choice ";
 cin >> choice;
 }
}

```

switch (choice)

case 1:

`cout << "sum" << m1 + m2 << endl;`

`break;`

case 2:

`cout << "Multiplication" << m1 * m2 << endl;`

`Break`

case 3

`cout << "Division" << (m1 / m2) << endl;`

`Break`

case 4

`cout << "Subtraction = " << m1 - m2 << endl;`

`Break`

case 5

`cout << "Square root of " << m1 << " = "`

`<< sqrt(m1) << endl;`

`cout << "Square root of " << m2 << " = "`

`<< sqrt(m2) << endl;`

`Break`

case 6:

~~`cout << m1 << "is " << (m1 * 100.00) / m2)`~~

~~`<< "% of " << m2 << endl;`~~

`Break`

case 7

~~`cout << "Square of " << m1 << " = " << pow(m1,`~~

~~`cout << "Square of " << m2 << " = " << pow(m2,`~~

`Break`

case 8

~~`cout << "sin(" << m1 << sin(m1);`~~

~~`cout << "sin(" << m2 << sin(m2);`~~

## 1) Standard I/O

Break)

default

cout << "Invalid choice" << endl; //

{

{

int main()

{

A < double > obj;

obj.accept();

obj.decalc();

return 0;

}

; "less string" > true  
(1). ~~free, (a), slity~~  
[2/1]

(1). travel, a > true

(2). fed, b

; line > true

(1). anim, tri

; d, tri

; mon, privet

; semaf > privet > every

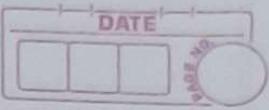
(1). slity

18. n / et nopol3 Mi. fgp. s. n / deff. 1 > true  
17. n / et nopol3 Mi. fgp. s. n / deff. 1 > true  
16. n / et nopol3 Mi. fgp. s. n / et nopol3 enq  
15. n / et nopol3 Mi. fgp. s. n / et nopol3 enq

## Experiment 11

```
1] #include <iostream>
#include <queue>
using namespace std;
void prints(queue<string> q)
{
 if (q.empty())
 cout << "Queue is Empty";
 else
 cout << "Queue Elements are:";
 while (!q.empty())
 cout << q.front();
 q.pop();
 cout << endl;
}

int main()
{
 int ch;
 string name;
 queue<string> games;
 while (1)
 cout << "1. Push\n2. Pop\n3. Display Queue\n4. Size\n5. Exit\n";
 cin >> ch;
 switch (ch)
 {
 case 1:
 cout << "Enter Name: ";
 cin >> name;
 games.push(name);
 break;
 case 2:
 cout << "Popped Element is: ";
 cout << games.front();
 games.pop();
 break;
 case 3:
 cout << "Display Queue: ";
 prints(games);
 break;
 case 4:
 cout << "Size of Queue is: ";
 cout << games.size();
 break;
 case 5:
 exit(0);
 }
}
```



```
cout << "Enter choice:\n";
cin >> ch;
switch(ch)
{
 case 1:
 cout << "Enter game Name:" << endl;
 cin >> name;
 games.push(name);
 break;
 case 2:
 cout << "Popped All Elements:\n\n";
 while (!games.empty())
 {
 cout << games.front() << endl;
 games.pop();
 }
 cout << endl;
 break;
 case 3:
 cout << "Popped One Elements:" << games.front();
 games.pop();
 break;
 case 4:
 cout << "Queue size :" << games.size();
 break;
 case 5:
 exit(0);
 break;
 default:
 cout << "Invalid choice :\n";
 break;
}
```

2] #include <iostream>  
#include <vector>  
#include <cctype>  
using namespace ~~type~~ std;  
int main ()  
{  
 vector<int> vec (5);  
 int i;  
 cout << "Enter Vector 5 Elements:";  
 for (i = 0; i < 5; i++)  
 cin >> vec [i];  
 cout << "Vector Elements are:" << endl;  
 for (i = 0; i < 5; i++)  
 cout << vec [i] << endl;  
 cout << "Modified Elements are:";  
 for (i = 0; i < 5; i++)  
 vec [i] = vec [i] + i \* 2;  
 for (i = 0; i < 5; i++)  
 cout << vec [i] << " ";  
 int scalar;  
 cout << "Enter a Scalar Value to Multiply:";  
 cin >> scalar;  
 cout << "After Multiplying by scalar:";

~~for (i=0; i < 5; i++)~~

~~vec[i] = vec[i] \* scalar;~~

~~for (i=0; i < 5; i++)~~

~~cout << vec[i] << endl;~~

~~return 0;~~

data type float [ ]

< min > obulai #

< max > obulai #

< data > obulai #

< Teeny > obulai #

data endl;

data arr < T > data

Qn : si(hu)

( ) tderas bior

ggis tri

"data arr .tab .arr .arr" > tri

ggis << arr

.stab T

(++i ; ggis > i ; o=i 'tri') asd

"data arr .tab .arr .arr" > tri

arr << arr

(stab) deuf' data - fun

( ) deuf' bior

at true now data .arr .arr .arr" > tri

line > a / stab

! n << arr

## Experiment : 12

### 1) Implement stack

```
#include <iostream>
#include <cstring>
#include <stack>
using namespace std;
template <class T>
class Stack
{
 stack <T> my_stack;
public:
 void accept()
 {
 int size;
 cout << "Enter the size of the stack ";
 cin >> size;
 T data;
 for(int i=0; i < size; i++)
 {
 cout << "Enter the data for the stack ";
 cin >> data;
 my_stack.push(data);
 }
 }
 void Pop()
 {
 cout << "Enter the value which you want to
 delete \n ";
 cin >> x;
```



```
If (my_stack.empty ())
 cout << "The stack is empty";

If (!My_stack.empty ())
 cout << "the stack is empty";

If (!My_stack.empty ())
 my_stack.pop (x);

void display ()
{
 If (My_stack.empty ())
 cout << "The stack is empty";

 while (!My_stack.empty ())
 cout << "My_stack.top ()";
 my_stack.pop ();

int main ()
{
 stack < int > k;
 k.accept ();
 k.display ();
```

## 2) Implement queue

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

template <class T>

```
class queue {
private
 Queue <T> q;
public:
 void Enqueue (T Element)
 {
 q.push (Element);
 cout << "Element " << "Enqueued Successfully";
 }
 void dequeue ()
 {
 if (q.empty ())
 cout << "Queue is Empty cannot dequeue";
 else
 cout << q.front () << "dequeued Successfully";
 q.pop ();
 }
 void display ()
 {
 cout << "Queue Elements : ";
 Queue <T> temp = q;
```

while (!temp.empty ())

cout < temp.front () << " ";  
 temp.pop ();

3  
6

int main ()

Queue<int> mq;  
 mq.enqueue(30);  
 mq.enqueue(20);  
 mq.enqueue(10);  
 mq.display();  
 mq.dequeue();  
 return 0;

Qn  
12/11

Home Devang Gadewar - devanggade

hackerrank.com/profile/devanggadewar

ChatGPT WhatsApp google classroom YouTube HiAnime GitHub

Search

0% Complete your profile  
Add your missing details →  
This data will be helpful to auto-fill your job applications

Devang Gadewar  
@devanggadewar

Personal Information

devanggadewar@gmail.com  
Add your mobile number  
India

My Badges

Problem Solving ★★★★  
C++ CPP ★★★★  
Sql SQL \*

My Resume + Add Resume  
Add your resume here

EEO settings

My Certifications

SQL (Intermediate) Verified SKILL

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

Privacy • Terms