

Experiment 1:

- q.1 Write a program to declare a class having data members as roll no. & name, accept & display data for one object.
- q.2 Write a program to declare a class group having data members as book name, price no of pages accept this data for 2 objects & display the name of book having greater price.
- q.3 Write a program to declare a class time. Accept the time in HH:MM:SS and convert it into total seconds & display them.

Programs:

```
1. #include <iostream>
```

```
using namespace std;
```

```
class student
```

```
{
```

```
    int roll;
```

```
    string name;
```

```
public:
```

```
    void accept()
```

```
{
```

```
    cout << "Enter the roll number of the student";
```

```
    cin >> roll;
```

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

```
    cin >> name;
```

```
}
```

```
    void display()
```

```
    cout << "The name of student is:" << name << endl;
```

```
    cout << "The roll no. of student is:" << roll << endl;
```

```
}
```

```

int main()
{
    student n2;
    n1.accept();
    cout << "Data" << endl;
    n1.display();
    return 0;
}
    
```

写于

* Output:

Enter roll number of the student: 13
 Enter the name of student : Abharva ~~Han~~
 Data:
 The name of student is : Abharva
 The roll no of student : 13

2)
→ #include <iostream>
using namespace std;

class book

{

 string name;

 float price;

 int pages;

 public:

 void accept();

}

cout << "enter the name of book";

cin >> name;

cout << "enter the price of the book";

cin >> price;

cout << "enter the no of pages";

cin >> pages;

}

 void display();

{

 cout << "Book Name = " << name;

 cout << " Book Price = " << price;

 cout << " Book pages = " << page;

}

}

 void main()

{

 Book b1, b2;

 b1.accept();

 b2.accept();

 b1.display();

 b2.display();

}

* 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] `#include <iostream>`
`using namespace std;`
`class time`

`{`
`int H, M, S, t;`

`public:`
`void accept()`

`{`

`cout << "Enter H,M,S";`
`cin >> H >> M >> S;`

`}`

`void display()`

`{`

`t = (H * 3600) + (M * 60) + S;`

`cout << "Total time in seconds:";`

`};`

`int main()`

`{`
~~time t;~~

~~t.accept();~~

~~t.convert();~~

* Output:

Enter time in HH.MM.SS Format: 1.30.45

Total time in seconds: 5445

Q
30/7/25

Experiment 2:

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

```
#include <iostream>
using namespace std;
class city
{
public:
```

```
string name;
int p;
```

```
void accept()
```

```
{
```

```
cout << "Enter the name of city";
```

```
cin >> name;
```

```
cout << "Enter the population of city";
```

```
cin >> p;
```

```
}
```

```
void display()
```

```
{
```

```
cout << "City:" << name << ", population" << p << endl;
```

```
}
```

```
,
```

```
int main()
```

```
{
```

```
city cities[5];
```

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

```
cout << "Enter details of city" << i+1 << "\n";
```

```
cities[i].accept(); }
```

```

int maxIndex = 0;
for(int i=1; i<5; i++)
{
    if(cities[i].p > cities[maxIndex].p)
    {
        maxIndex = i;
    }
}
cout << "City with the highest population is : "
<< cities[maxIndex].name;
cout << "(" << cities[maxIndex].p << "people)" << endl;

```

* Output:

Enter details of city 1

Enter name of city pune

Enter the population of city 1000

Enter details of city 2

Enter name of city mumbai

Enter the population of city 2000

Enter details of city 3

Enter name of city kokan

Enter the population of city 4000

Enter details of city 4

Enter name of city satara

Enter the population of city 82300

Enter details of city 5

Enter name of city Sangli

Enter the population of city 5600

City with the highest population is: Sangli(5600 people)

2] WAP to declare a class 'Account' having data members as account no. and balance, accept this data for 10 accounts & give interest of 10%. where balance is equal or greater than 5000 and display them.

```
#include<iostream>
```

```
using namespace std;
```

```
class account {
```

```
public:
```

```
    int accno;
```

```
    float balance;
```

```
    void accept();
```

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

```
    cin >> accno;
```

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

```
}
```

```
    void interest();
```

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

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

~~3~~~~void display();~~

```
    cout<<"Account No: "<<accno<<, Balance:"
```

```
    <<balance<<endl;
```

~~3~~~~3;~~

```
int main() {
```

```
    account acc[5];
```

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

```
        cout<<"Account "<< i+1 << ":" << endl;
```

```
        acc[i].accept();
```

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

```
        acc[i].interest();
```

cout << "Account details after applying interest.";
for (int i=0; i<5; i++) {
 acc[i].display();
}

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:" << endl;
```

```
}
```

```
int main()
```

```
staff s[5];
```

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

```
s[i].accept();
```

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

```
if (strcmp(s[i].post, "HOD") == 0)
```

```
s[i].display();
```

①
3017125

Experiment : 3

- a) Write a program to declare a class 'book' containing data members as book - title, author - name, price. Accept & display the information for one object.

→
#include<iostream>

using namespace std;
class book;
{

private:

int price;
char title[20];
char author[20];
public:

void accept()

{

cout << "Enter the price of book\n";
cin >> price;
cout << "Enter the title of book\n";
cin >> title;
cout << "Enter the author of book\n";
cin >> author;
}

void display()

{

cout << "\n Price : " << price;
cout << "\n Title : " << title;
cout << "\n Author : " << author;

}

};

FILE NO.		
DATE	/ /	

int main()

{

book b1;

book * p;

p = &b1;

p → accept();

p → display();

3

← GU

b] WAP to declare a class 'student' having data members roll_no & percentage. Using 'this' pointer invoke member functions to accept & display this data for one object of the class
 →

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

{

private:

```
int roll;
float per;
```

public:

void accept()

{

cout << "Enter the Roll number of the student:";

cin >> this->roll;

cout << "Enter the Percentage of the student:";

cin >> this->per;

}

void display()

{

cout << "Roll:" << this->roll << endl;

cout << "Percentage:" << this->per << endl;

}

int main()

{

student s1;

s1.accept();

s1.display();

}

Q) Write a program to demonstrate the use of nested class

→
#include <iostream.h>
using namespace std;
class student

{

public:

 class marks

{

private:

 int roll;

 float per;

public:

void accept()

{

 cout << "Enter the roll number of student:";
 cin >> roll;

 cout << "Enter the percentage of student:";

 cin >> per;

}

void display() {

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

 cout << "Percentage:" << per << endl;

}

}

}

int main()

{

 student :: marks m1;

 m1.accept();

 m1.display();

}

Ques

6/8/25

Experiment 4:

a] #include <iostream>
Using namespace std;
class result2;

class result1
{

int m1;
public:

void accept1()
{

cout << "Enter the mark of M1 = ";

cin >> m1;

}

friend void calculate(result1 p, result2 r);

};

class result2

{

int m2;

public:

void accept2()
{

cout << "Enter the marks of M2 = ";

cin >> m2;

}

friend void calculate(result1 p, result2 r);

};

void calculate(result1 p, result2 r);

};

void calculate(result1 p, result2 r)

{

int total = (p.m1 + r.m2) / 2;

cout << "Average of the result := " << total << endl;

}

int main()

{

result1 p;

result2 r;

p.accept1();

r.accept2();

calculate(p, r);

}

b) #include <iostream>

using namespace std;

class number2;

class number1 {

{

int num1;

public:

void accept1()

{

cout << "Enter the number 1 = ";

cin >> num1;

}

friend void calculate(number1 p, number2 r);

};

class number2

{

int num2;

public:

void accept2()

{

cout << "Enter the number 2 = ";

cin >> num2;

}

friend void calculate(number1 p, number2 r);

};

```
void calculate(number1 p, number2 r)
```

```
{
```

```
if (p.num1 > r.num2)
```

```
{
```

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

```
}
```

```
else {
```

```
cout << "The greatest number is :" << r.num2 << endl;
```

```
}
```

```
int main()
```

```
{
```

```
number1 p;
```

```
number2 r;
```

```
p.accept1();
```

```
r.accept2();
```

```
calculate(p, r);
```

```
}
```

c) Swapping 2 no. without using friend function

```
#include<iostream>
using namespace std;
class demo {
public:
```

```
int p, q;
```

```
void accept() {
```

```
cout << "Enter 2 numbers: " << 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();
```

```
}
```

d) Swapping 2 numbers from same class by using friend function.

→

```
#include <iostream>
using namespace std;
class demo {
    int a, b;
public:
    void accept() {
        cout << "Enter 2 numbers:" << endl;
        cin >> a >> b;
    }
}
```

```
void display() {
    cout << "Value of a:" << a;
    cout << "Value of b:" << b;
}
```

```
friend void swapnums(demo &t);
```

}

~~Void swapnums(demo &t) {~~

```
int temp = t.a;
t.a = t.b;
t.b = temp;
```

}

```
int main() {
    demo k;
    k.accept();
    swapnums(k);
    k.display();
}
```

e) 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 swapnumbers(A &p, B &q);
```

```
};
```

```
class B {
```

```
int b;
```

```
public:
```

```
void accept2() {
```

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

```
cin >> b;
```

```
}
```

```
void display2() {
```

```
cout << "Value of b:" << b;
```

```
}
```

```
friend void swapnumbers(A &p, B &q);
```

```
};
```

void swapnumbers(A &p, B &q) { }

int temp = p.a;

p.a = q.b;

q.b = temp;

}

int main() {

A k;

B f;

k.accept();

f.accept();

swapnumbers(k, f);

k.display1();

f.display2();

}

Q1.

→ #include<iostream>

using namespace std;
class sum

{

private:

int num, summy = 0;

public:

sum()

{

num = 5;

for (int i = 0; i < num; i++)

{

summy = summy + num;

{

cout << "Sum of n number is" << summy << endl;

{

sum(int n) {

num = n;

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

summy = summy + num;

{

cout << "Sum of n number is" << summy << endl;

{

sum(sum &t) {

num = t - num;

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

summy = summy + num;

{

cout << "Sum of n number is" << summy << endl;

{

{;

int main ()

{

sum k;

sum z (3h);

sum x (z);

}

Pen

18/9/25

Q2. Code for class 'student' having data members
as name & percentage.
→

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

```
class student
{
```

```
    float per;
    string name;
public:
```

```
student()
```

```
{
```

```
cout << "Enter Name : ";
```

```
cin >> name;
```

```
cout << "Enter the Percentage : ";
```

```
cin >> per;
```

```
}
```

```
void display()
```

```
{
```

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

```
cout << "Percentage : " << per << endl;
```

```
}
```

```
};
```

```
int main()
```

```
{
```

```
student s1;
```

```
s1.display();
```

```
}
```

NAME		
DATE	/ /	

* Out put:

→

Enter Name: Atharva

Enter the Percentage : 80

Name : Atharva

Percentage : 80

Q3. Define class "collage" data members as roll no., name, course. Accept & display data for 2 students.

→

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

```
class collage
{
```

```
    int roll;
    string name, course;
public:
```

```
    collage (int r, string n)
    {
```

```
        roll = r;
        name = n;
        course = "Computer Science";
    }
```

```
    void display()
```

```
{
```

```
    cout << "The name of student = " << name << endl;
```

```
    cout << "The roll number of student = " << roll << endl;
```

```
    cout << "The course of student = " << course << endl;
```

```
}
```

```
};
```

DATE	/ /
------	-----

int main()

{

 collage s1(11, "Atharva"), s2(10, "Aditya");
 s1.display();
 s2.display();

}

* Output

→

The name of student = Atharva

The roll number of student = 11

The course of student = computer science

The name of student = Aditya

The roll number of student = 10

The course of student = computer science

Q4. Constructor over loading.

→

#include <iostream>
using namespace std;

class Rectangle;

{
int l, w;
public:

rectangle()
l = 1;
w = 2;
}

rectangle(int a)

{
l = a;
w = a;
}

rectangle(int a, int b)

{
l = a;
w = b;
}

void calculate()

{

int a;
a = l * b;

cout << "Area = " << a;

{

};

int main()

{

rectangle r1;

rectangle r2(5);

rectangle r3(4, 5);

r1.calculate();

r2.calculate();

r3.calculate();

}

* Experiment No. : 6

Q1. 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 << endl;

cout << "Age: " << age << endl;

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

}

,

int main()

{

~~student s1;~~

s1.accept();

s1.display();

Q2. 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 & sports~~~~cin >> marks >> s-score;~~~~{~~~~void calculate()~~~~{~~
$$total = (marks + s-score) / 200 \times 100$$
~~{~~

FAIL NO.	
DATE	/ /

void display()

{

cout << "Marks : " << marks;

cout << "Sports score : " << s-score;

cout << "Total score : " << total;

}

}

int main()

{

Result R1;

R1.accept();

~~R1.calculate();~~

R1.display();

}

Q3. 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 & battery capacity";
```

```
cin >> brand >> model >> type >> battery;
```

```
{
```

```
void display()
```

~~cout << "Brand name: " << brand << endl;~~~~cout << "Model name: " << model << endl;~~~~cout << "Type: " << type << endl;~~~~cout << "Battery capacity: " << battery << "WH";~~

```
{
```

```
}
```

int main()

{

EV e1;

e1.accept();

e1.display();

}

d] 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 & dept:";
```

```
cin >> name >> emp-id >> dept;
```

```
}
```

```
void display()
```

```
{
```

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

```
cout << "Emp-id:" << emp-id << endl;
```

```
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 out.output()
```

```
{
```

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

```
cout << "Emp-id :" << emp_id << endl;
```

```
cout << "Dept :" << dept;
```

```
    
```

```
};
```

```
int main()
```

```
{
```

~~Manager m;~~~~m.accept();~~~~m.display();~~

```
Developer d;
```

```
d.input();
```

```
d.output();
```

```
return 0;
```

```
}
```

e] Hybrid:

→

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

```
class Person
```

{

protected:

```
string name;
```

```
int age;
```

?;

class

```
class student : protected Person
```

{

protected:

```
int roll;
```

```
Academics aca;
```

?;

```
class Academics
```

{

public:

```
int marks;
```

?;

```
class Sports
```

{

public:

```
int sScore;
```

?;

class Result : public sports, public student

{

private:

int total;

public:

void accept()

{

cout << "Enter name & age";

cin >> name >> age;

cout << "Enter Roll no.:";

cin >> roll;

cout << "Enter academic & sports marks:";

cin >> aca_marks >> s_score;

}

void calculate()

{

total = (aca_marks + s_score) / 2;

}

void display()

{

cout << "Name:" << name << "age:" << age;

~~cout << "Roll no.:" << roll << "Aca. marks:" << aca_marks;~~

~~cout << "Sports score:" << s_score << "Total:" << total;~~

} } } ;

int main()

{

Result r;

r.accept();

r.calculate();

r.display();

}

f) #include <iostream>
using namespace std;
class clg-student
{

protected:
string clgcode;
};
class Test : virtual public clg-student
{

protected:
int perc;
};
class sports : virtual public clg-student
{

protected:
int s-score;
};
class Result : protected Test, sports
{

public:
void accept()
{

cout << "Enter clgcode, perc, s-score;"
cin >> clgcode >> perc >> s-score;
}

void display()
{

cout << "clgcode = " << clgcode;
cout << "In percentage = " << perc;
cout << "In sports score = " << s-score;
}
};

1	2	3	4
LATE		/ /	

```
int main()
```

```
{
```

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

Qn

26/9/25

*Experiment - 7 *

- a) Write a program using function overloading
 calculate area of laboratory and circle

```
#include <iostream>
```

```
using namespace std;
```

```
class area1
```

```
{
```

```
public:
```

```
int l, b;
```

```
void area1(int a, int b)
```

```
{
```

```
int c = a * b;
```

```
cout << "Laboratory area: " << c << "sq";
```

```
}
```

```
void area1(int s)
```

```
{
```

```
int f = s * s;
```

```
cout << "class area: " << f << "sq";
```

```
}
```

```
};
```

```
int main()
```

```
{
```

```
area1 m;
```

```
m. area1(20, 30);
```

```
cout << endl;
```

```
m. area1(20);
```

```
}
```

b) Write a program using function overloading to calculate the sum of 5 float values & sum of 10 integer values

→

```
#include<iostream>
```

```
using namespace std;
```

```
class sum1
```

```
{
```

```
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 numbers:" << s << endl;

```
{
```

```
void sum(int b[10])
```

```
{
```

```
int s=0;
```

```
for(i=0; i<10; i++)
```

```
{
```

```
s += b[i];
```

```
}
```

cout << "Sum of 10 integer numbers:" << s << endl;

```
{
```

```
};
```

```
int main()
{
```

```
    sum1 s1;
    float c[5];
    int d[10];
```

```
    cout << "Enter 5 float numbers: \n";
    for (int i = 0; i < 5; i++)
```

```
{
```

```
    cin >> d[i];
```

```
}
```

```
    cout << "Enter 10 integer numbers: \n";
```

```
    for (int i = 0; i < 10; i++)
```

```
{
```

```
    cin >> d[i];
```

```
}
```

```
    s1 = sum(c);
```

```
    s1 = sum(d);
```

```
}
```

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

→

```
#include <iostream>
using namespace std;
class num
{
    int a;
public:
    void accept()
    {
        cout << "enter value of a: ";
        cin >> a;
    }
}
```

```
void display()
{
    cout << "value of a:" << a;
}
```

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

```
int main()
```

```
{
    num n1;
```

```
n1.accept();
-n1
```

```
n1.display();
}
```

a) WAP to implement the unary ++ operator when used with the object so that numeric data member of 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;
```

```
}
```

```
    void display()
```

```
{
```

```
    cout << "value of a:" << a;
```

```
}
```

```
    void operator++()
```

```
{
```

```
    a = ++a;
```

```
}
```

```
};
```

```
int main()
```

```
{
```

```
    num n1;
```

```
    n1.accept();
```

```
    ++n1;
```

~~```
n1.accept();
```~~

```
n1.display();
```

```
}
```

~~Ques~~  
~~still~~

## \* Experiment 8 \*

Q1]

→

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

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

2.

→ #include<iostream>

Using namespace std;

class ilogin

{

protected:

string name;

string pass;

public:

virtual void accept()

{

cout << "Enter Name & password"

cin >> name >> pass;

}

virtual void display()

{

cout << "Name = " << name << "password" << pass;

{

};

class elogin: Public login

{

string email;

string pass;

public:

void accept()

{

cout << "Enter .email & password:";

cin >> email >> pass;

{

\*edit

void display()

{

cout << "Email " << email << "password " & pass;

}

};

class mlong : public ilogin

{

string mid;

string pass;

public:

void accept()

{

cout << "Enter M-id and password";

cin >> mid >> pass;

}

cout << "M-id " << mid << "password " & pass;

}

};

int main()

{

ilogin \* iptr

elogin i;

elogin e;

mlogin m;

iptr = i;

iptr → accept();

iptr → disp();

cout << endl;

iptr = & e;

iptr → disp();

cout << endl;

return 0;

}

Ques  
5/11

## Experiment 9:

```

→ #include <iostream>
#include <fstream>
using namespace std;
int main()
{
 ifstream fin;
 ofstream fout;
 fout.open("destination.txt");
 fin.open("source.txt");
 if (!fin)
 {
 cout << "error opening source file \n";
 return 1;
 }
 char ch;
 while (fin.get(ch))
 {
 fout.put(ch);
 }
 fin.close();
 fout.close();
 cout << "file opened successfully \n";
 fin.open("source.txt");
 int wordCount = 0;
 string word;
 while (fin >> word)
 {
 wordCount++;
 }
}

```

|          |     |
|----------|-----|
| FILE No. | / / |
| DATE     | / / |

```

cout << "Word count;" << wordCount << "\n";
fin.close();
fin.open("source.txt");
string target;
int count = 0;
while (fin >> word) {
 if (word == target) {
 count++;
 }
}

```

```

cout << "Enter target" << endl;
cin >> target;
cout << "Word occurrence" << count << endl;
fin.close();

```

```

fin.open("source.txt");
int digitCount;
int spaceCount;

```

```

while (fin.get(ch)) {
 if (isDigit(ch)) {
 digitCount++;
 }
}

```

```

if (isspace(ch)) {
 spaceCount++;
}

```

```

if (isspace(ch)) {
 spaceCount++;
}

```

```

}

```

cout << "Digit :" << digitCount << endl;  
~~cout << count << "space :" << spaceCount << endl;~~  
return 0;  
}

int main()

{

login \* login;

{

mail login e;

Membership login m;

login = & e

login → accept();

login → display();

return 0;

}

Q  
X

## Experiment 10

Q).

```
→ #include<iostream>
#include <math>
Using namespace std;
```

```
template<class T>
class A
{
public:
 T m1, m2;
```

```
void accept()
```

```
cout << "Enter the first number : ";
cin >> m1;
```

```
cout << "Enter the second number : ";
cin >> m2;
```

```
}
```

```
void calc()
```

```
int choice:
```

```
cout << "\n -- 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 * 100.0) / m2) << ".0"
<< endl;
```

Break;

case 7:

```
cout << "square of " << m1 << " = " << pow(m1, 2)
<< endl;
```

```
cout << "square of " << m2 << " = " << pow(m2, 2) << endl;
Break;
```

Case 8:

```
cout << "sin(\"m1\") = " << sin(m1) << endl;  
cout << "sin(\"m2\") = " << sin(m2) << endl;  
Break;
```

default:

```
cout << "Invalid choice!" << endl;
```

int main()

```
{  
    A<double> obj;  
    obj.accept();  
    obj.calculate();  
    return 0;  
}
```

Experiment 11

```
→ #include <iostream>
#include <vector>
#include <cc type>
using namespace std;
int main()
{
    vector<int> vec(5);
    int i;
    cout << "Enter vector 5 elements : ";
    for(i=0; i<5; i++)
    {
        cin >> vec[i];
    }
    cout << endl;
    cout << "vector elements are : " << endl;
    for(i=0; i<5; i++)
    {
        cout << "Modified Elements are : ";
        vec[i] = vec[i] + i * 2;
    }
    cout << endl;
    for(i=0; i<5; i++)
    {
        cout << vec[i] << " ";
    }
    cout << endl;
    int scalar;
    cout << "Enter a scalar value to multiply ";
    cin >> scalar;
    cout << "After Multiplying by scalar : ";
}
```

| | |
|----------|-----|
| Page No. | / / |
| Date | |

```
for(i=0; i<5; i++) {  
    vec[i] = vec[i] * scalar;  
}
```

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

Experiment 12

1)

```
#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() {
        T a;
        cout << "Enter the value which you want to delete\n";
        cin >> x;
        if (my_stack.empty()) {
            cout << "The stack is empty\n";
        } else if (!my_stack.empty()) {
            cout << "The stack is not empty\n";
        }
    }
};
```

| | |
|------|-----|
| NAME | |
| DATE | / / |

cout << "The stack is empty" << endl;

}
if (!MyStack.empty()) {
myStack.pop(x);
}

void display() {

if (myStack.empty()) {

myStack.pop(x); cout << "The stack is empty"
} } } } } } }

while (!MyStack.empty()) {

cout << myStack.top();

myStack.pop();

}

}

int main(void) {

stack<int> k;

k.accept();

k.display();

}

2] 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" << endl;

}

void dequeue() {

if(q.empty()) {

cout << "Queue is Empty cannot dequeue" << endl;

q.pop();

}

}

void display() {

cout << "Queue Elements : " << endl;

queue<T> temp = q;

while(!temp.empty()) {

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

temp.pop();

}

cout << endl;

}

};

Final Exam

```
int main() {  
    Queue<int> mq;  
    mq.Enqueue(30);  
    mq.Enqueue(20);  
    mq.Enqueue(10);  
    mq.Display();  
    mq.Dequeue();  
    mq.Display();  
  
    return 0;  
}
```

Open
Stack

Experiment 11

```

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

template <typename T>
class generic_vector {
    vector<T> data;
public:
    void create(int n) {
        data.resize(n);
        cout << "Enter: ";
        for (int i = 0; i < n; i++) {
            cin >> data[i];
        }
    }

    void modify(int index, T value) {
        if (index >= 0 && index < data.size())
            data[index] = value;
    }

    void multiply(T scalar) {
        for (auto &el : data)
            el *= scalar;
    }

    void display() {
        cout << "Vector: ";
        for (const auto &el : data)
            cout << el << " ";
        cout << endl;
    }
};
```

```

int main()
{
    generic_vector<int> vec;
    int n;
    cout << "Enter no. of elements : ";
    cin >> n;
    vec.create(n);
    vec.display();
    cout << "Modify elements at index 1 to
           20 : " << endl;
    vec.modify(1, 20);
    vec.display();
}

```

```

cout << "Multiply all elements by 2 : " << endl;
vec.multiply(2);
vec.display();

```

return 0;

}

* Output: Enter number of elements: 3
~~Enter 3 elements:~~

10 15 30

vector: 10 15 30

Modify element at index 1 to 20:

vector: 10 20 30

Multiply all elements by 2:

vector: 20 40 60