Experiment : 1.

1. WAP to declare a class student having data numbers as name, roll no. Accept & display data for one student

```
#include<iostream>
using namespace std;
class student {
    string name;
    int roll_no;
  public:
    void accept();
    void display();
};
void student :: accept () {
  cout << "Enter student name & roll no:";
  cin >> name >> roll_no;
}
void student :: display () {
  cout << "\n student name:" << name;
  cout << "\n student roll no: " << roll_no;
}
int main() {
  Student SI;
  SI.accept();
  SI.display();
  return 0;
}
```

Output:-

```
Enter name & roll no: abc
22
student name: abc
Student roll no: 22.
```

2. WAP to declare a class book, having data members as id, name, price. Accept data for 2 books & display data of book having greater price.

→
```
#include <std iostream>
using namespace std;
class book {
 string name;
 int id, price;
 public:
   void accept();
   void display();
   int get_price();
};
                accept
void book :: display() {
cout<<"Enter name, price & id of book:";
cin>>name>>price>>id;
}
void book :: display() {
cout<<"\n Book name:"<<name;
cout<<"\n Book price:"<<price;
cout<<"\n Book id:"<<id;
}
 int book::get_price(){
 return price;
}
int book main() {
  book b1, b2;
cout<<"Enter details for book 1:\n";
b1.accept();
cout<<"Enter details for book 2:\n";
b2.accept();
cout<<"\n Details of book1:";
b1.display();
cout<<"\n Details of book:";
b2.display();
cout<<"\n\n Book with greater price:\n";
if (b1.get_price() > b2.get_price()) {
   b1.display();
} else if (b2.get_price()> b1.get_price()) {
   b2.display();
} else {
     cout<<"Both books have same price";
}
return 0;
}
```

```
void greatest(classA a, classB b){
 cout << "Greatest number:";
  if (a.num1 > b.num2)
    cout << a.num1;
    else
    cout << b.num2;
}

int main (){
 classA a;
 classB b;
 a.accept();
 b.accept();
  greatest (a,b);
 return0;
}
```

Output

Enter 1st no. = 12

Enter 2nd no. = 14

Greatest number = 14.

(c) WAP for swapping contents of two variables of same class using friend function

```
#include<iostream>
using namespace std;
class swap(){
 int value;
public:
    void accept (){
      cout<<"Enter value:";
      cin>> value;
     }
    void disp(){
     cout<<value<<;
    }
   friend void swapnum (swap &, swap&);
};
void swap-num (swap &x, swap &y){
 int temp = x.value;
  int x.value = y.value;
      y.value = temp;
}
int main(){
swap obj1, obj2;
obj1.accept();
obj2.accept();
cout<<"\n Before swapping";
```

Output:-

```
Enter details for book 1:
Enter name, price and id of book: abc
100
1234

Enter details for book 2:
Enter name, price & id of book: xyz
200
5678

Details of book 1:
Book name: abc
Book price: 100
Book id: 1234
Details of book 2:
Book name: xyz
book price: 200
Book id: 5678
Book with greater price:-
Bookname: xyz
Book price: 200
Book id: 5678
```

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

→

```
#include <iostream>
using namespace std;
class Time {
  int H, M, S;
public:
  void accept();
  void display();
};
void Time :: accept () {
  cout << "Enter time in hours, minutes & seconds:";
  cin >> H >> M >> S;
}
void Time :: displayTotal() {
  int totalseconds = H*3600 + M*60 + S;
  cout << "\nTotal time in seconds:" << totalSeconds
}
int main () {
Time t1;
t1.accept();
t2.display();
return 0;
}
```

Output:

Enter time in hours, minutes & seconds: 3
22
10
Total time in seconds 12130.

15/7

Experiment 2.

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

→ Code:-

```
#include<iostream>
using namespace std;
class city {
public:
    string name;
    int ~~population~~, pop;
};
int main(){
 city cities[5];
  for(int i=0; i<5; i++){
 cout<<"Enter name & population of city"<<i+1<<";
cin>> cities[i].name >> cities[i].pop;
}
int maxIndex=0;
for(int i=0; i<5; i++){
  if(cities[i].pop >> cities ~~[i].pop~~[maxIndex].pop)
     maxIndex = i;
}
cout<<"City with highest population:"<<cities[maxIndex].name
return 0;
}
```

Output:-

Enter name & Population of city 1 : Brazil
220
Enter name & population of city 2 : Morocco
350
Enter name & population of city 3 : London
60
Enter name & population of city 4 : Ibiza
600
Enter name & population of city 5 :
430
city with highest population : Ibiza

6) WAP to declare a class 'Account' having data members as account no. & balance. Accept this data for 10 accounts & given interest of 10% where balace is equal or greater than 5000 and display them.

→ code:-

```
#include <iostream>
using namespace std;
class account{
public:
  int acc_no, balace;
};
int main(){
 account acc[10];
 for(int i=0; i<10; i++){
cout<<"Enter acc no. & balance for acc"<<i+1;
 cin>> acc[i].acc_no>> acc[i].balance;
}
cout<<"Accounts with interest (10% if balance>=5000)
 for(int i=0; i<10; i++){
   if(acc[i].balance >=5000){
     double interest = acc[i].balance*0.10;
cout<<"Acc No:"<< acc[i].acc_no<< "Interest
     << acc[i].acc_no<< interest << endl;
  }
}
return 0;
}
```

O/P:-

Enter acc no & balance of acc 1: 22
400
Enter acc no & balance of acc 2: 11
200
Enter acc no. & balance of acc 3: 33
600
Enter acc no & balance of acc 4: 77
300
Enter acc no. & balance of acc 5: 44
100
Enter acc no. & balance of acc 6: 55
900
Enter acc no. & balance of acc 7: 66
350
Enter acc no. & balance of acc 8: 99
500
Enter acc no. & balance of acc 9: 88
800
Enter acc no. & balance of acc 10: 98
24567

Accounts with interest applied (10% if balance >= 5000)
Account No. 98, Interest: 2456.7.

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

→
```
#include<iostream>
using namespace std;
class staff{
 public:
   string name, post;
};
int main(){
staff stf[5];
for (int i=0; i<5; i++){
    cout<<"Enter name & post of staff: "<<i+1;
    cin>>stf[i].name>>stf[i].post;
}
cout<<"staff members who are HOD: \n";
    for (int i=0; i<5; i++){
      if (stf[i].post == "HOD")
    cout<<stf[i].name<<endl;
}
return 0;
}
```

Output:-

Enter name & post of staff 1: sam
HR
Enter name & post of staff 2: cam
CEO
Enter name & post of staff 3: jam
Cameraman.
Enter name & post of staff 4: pam
Singer
Enter name & post of staff 5: zam
media
~~Enter name & post of staff 6:~~

staff members who are HOD:-

EXPERIMENT-3

DATE 25 07 25

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 using a pointer to that object.

→

```
#include <iostream>
using namespace std;
class book{
 string name, author;
 int price;
 public:
 void accept(){
 cout<<"Enter book name:";
 cin >> name;
 cout << "\n Enter Author's name:";
 cin>> author;
 cout << "\n Enter the price";
 cin>> price;
}
void disp(){
cout<<"\nBook name: "<< name; this->name;
cout <<"\n Book author:" << author; this->author;
cout<<"\n Book price :" << price; this->price;
}
};
```

```
int main(){
book* b = new book;
b-> accept();
b-> disp();
 delete b;
 return 0;
};
```

O/P

Enter book name : The Alchemist
Enter author name : Paulo Coelho
Enter price : 399

Book name : The Alchemist
Book author : Paulo Coelho
Book price : 399.

B) WAP to declare a class 'student', having data members roll no & percentage. Using 'this' pointer invoke member functions to accept and display data for one object of the class.

→

```
#include <iostream>
using namespace std;
class student {
  int roll-no, percen;
 public:
  void accept() {
 cout<< "Enter roll no:";
 cin >> roll-no; this-> roll-no;
 cout << "Enter percentage:";
 cin >> this-> percen;
}
void disp() {
 cout <<"\n Student Information:";
 cout << "\n Roll No.:" << this-> roll-no;
 cout << "\n Percentage:" << this -> percen;
}};
int main(){
Student *s = new student;
s-> accept();
s-> disp();
```

delete s;
return 0;
}

O/P:-

Enter roll no: 45
Enter percentage: 89

Student Information:
Roll no: 45
Percentage: 89%.

19/9

EXPERIMENT 4

A) WAP to create two classes result1 & result2 which store marks of students. Read the values of marks for both classes objects & compute the average of 2 results.

→
```
#include<iostream>
using namespace std;
class result1{
  int marks;
  void accept(){
   cout<<"Enter marks for result 1:";
   cin>>marks;
  }
};
class result2{
  int marks;
   void accept(){
  cout<<"Enter marks for result 2:";
  cin>>marks;
}
};
class result{
 float avg;
 void average(result1 R1, result2 R2){
  avg = (R1.marks + R2.marks)/2.0;
}
```

```
void disp(){
 cout<<" Average marks" << avg ;
 }
};
int main(){
 result1 R1;
 result2 R2;
 result  R;
 R1.accept();
 R2.accept();
 R.average();
 R.disp();
 return 0;
}
```

Output:-

Enter marks for Result 1 : 38.
Enter marks for Result 2 : 40.
Average ~~marg~~ marks : 39.

b. WAP to find the greatest number among two numbers from two different classes using friend function.

→

```
#include<iostream>
using namespace std;
~~classB;~~ class classB;
class classA{
  int num1;
  ~~void~~ public:
      void accept(){
    cout<<"Enter 1st no.:";
    cin>>num1;
  }
  friend void ~~average~~ greatest (classA, classB);
};
class classB {
  int num2;
  public:
    void ~~accp~~ accept(){
    cout<<"Enter 2nd number:";
    cin>>num2;
  }
  friend void greatest (classA, classB)
};
```

```
cout << "obj1 = "; obj1.disp();
cout << "obj2 = "; obj2.disp();
swap-num (obj1, obj2);
cout << "In After swapping :";
cout << "obj1 = "; obj1.disp();
cout << "obj2 = "; obj2.disp();
return 0;
}
```

Output :-

Enter value : 2

Enter value : 3.

Before swapping

obj1 = 2

obj2 = 3.

After swapping :

obj1 = 3.

obj2 = 2.

d. WAP to swap numbers from same class using object as function argument. Write swap function as member function

→

```
#include <iostream>.
using namespace std;
class swapClass {.
 int value;
public:
  void accept(){
   cout << "Enter value";
   cin >> value;
  }
  void disp(){.
    cout << value;
}
void swapvalues (swapclass &obj){.
   int temp = value;
     value = obj.value;
     obj.value = temp;
  }
};
int main(){
swapclass obj1, obj2;
obj1.accept();
obj2.accept();
```

```
cout << "\n Before swapping";
cout << " obj1 = "; obj1.disp();
cout << " obj2 = "; obj2.disp();

obj1.swapvalues (obj2);

cout << "\n After swapping: ";
cout << " obj1 = "; obj1.disp();
cout << " obj2 = "; obj2.disp();
return 0;
}
```

O/P:-

```
Enter value : 10
Enter value : 20
Before swapping:
 obj1 = 10
 obj 2 = 20
After swapping:
 obj1 = 20
 obj 2 = 10.
```

e. WAP to swap 2 no.s from different class using friend function.

→

```
#include <iostream>
using namespace std;
class B; class classB;
class classA {
 int num1;
public:
 void accept () {
  cout << " Enter 1st number :";
  cin >> num1;
 }
 void disp() {
  cout << " No in c
 friend void swapValues (classA &, classB &);
};
class classB {
int num2;
public:
 void accept() {
  cout << " Enter 2nd number ";
  cin >> num2;
 }
 friend void swapValues (classA &, classB &);
};
void swapValues (classA &a, classB &b) {
```

```
int temp = a.num1;
a.num1 = b.num2;
b.num2 = temp;
}
int main(){
classA objA;
class B objB;
objA.accept();
obj.accept();

cout<<"\n Before swapping";
~~cout<<"obj1"<<~~
objA.disp();
obj B.disp();
cout<<" After swapping";
objA.disp();
obj B.disp();
return 0;
}
```

O/P :-

Enter 1st no. = 10

Enter 2nd no. = 20

Before swapping = 10, 20

After swapping = 20, 10.

1919

Experiment 5.

DATE 05/09/25

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

→

```
#include <iostream>
using namespace std;
class num {
  int n,sum;
  Public:
    num(){
      n=5;
      sum=0;
    }
    num (int x){
      n=x;
      sum=0;
    }
    void s(){
      for(int i=1; i<=n; i++){
        ~~sum=~~ sum+=i;
      }
    }
    void disp(){
      cout<<"SUM:"<<sum<<"\n";
    }
};
```

```
int main(){
  num N1;
  num N2(10);
  N1.S();
  N1.disp();
  N2.S();
  N2.disp();
 return 0;
};
```

O/P:-

SUM: 15

SUM: 55

b. Write a program to declare a class "student", having data members as name & percentage. Write a constructor to initialize these data members. Accept & display data for one student.

→

```
#include <iostream>
using namespace std;
class student {
  ~~int~~ string name;
      float per;
  Public:
    student (string s, float p){
      name = s;
      per = p;
    }
    void disp(){
    cout<<" Name: "<< name <<"\n";
    cout << "Percentage: "<< per<<~~"\n"~~; <<"%";
    }
};
int main(){
student s1("ABC", 67);
 s1.disp();
 return 0;
};
```

O/P:-

Name: ABC

Percentage: 67%

c. define a class 'college' members variables as roll-no, name, course. WAP using constructor with default value as "computer Engineering" for course. Accept this data for two objects of class & display the data.

→
```
# include <iostream>
#include<string>
using namespace std;
class college{
  int roll-no;
  string name;
  string course;
Public:
  college(){
   roll-no = 101;
   name = "ABC";
   course = "computer Engineering";
  }
  college(int r, string n, string c = "computer Engineering"){
   roll-no = r;
   name = n;
   course = c;
  }
  void disp(){
cout<< " Name:" << name << "\n";
cout<< " Roll NO:" << roll-no << "\n";
cout << "course" << course;
  }
};
int main(){
student s1;
student s2(102, "XYZ");
  s1. college c1;
      college c2(102, "XYZ");
   c1.disp();
   c2.disp();
  return 0;
}
```

O/P :-

Name : ABC.
Roll NO : 101.
course : Computer Engineering

Name : XYZ.
Roll NO : 102
course : Computer Engineering.

constructor overloading.

d.
```
#include<iostream>
using namespace std;
class add {
int a, b, sum;
Public:
  add (){
   a = 50;
   b = 60;
   sum = a+b;
   cout << "Sum=" << sum << "\n";
  }
  add (int x) {
   a = x;
   b = x;
   sum = a+b;
  cout << "sum=" << sum << "\n";
  }
  add (int x, int y) {
   a = x;
   b = y;
  sum = a+b;
  cout << "sum=" << sum << "\n";
  }
};
int main ()
add A1;
add A2 (40);
add A3 (20, 30);
return 0;
}
```

O/P:-

sum = 110
sum = 80
sum = 50

1919

Experiment 6.

a) Write a program to implement multilevel inheritance. Assume suitable

→
```
#include <iostream>
using namespace std;
class Dept {
protected :
    string dname;
};
class student : protected dept {
 string sname;
 int roll-no;
};
class marks : protected dept, protected student {
 int m1, m2;
 float per; Public: void cal() {
cout << " Enter Department name:" << endl;
cin >> dname;
cout << " Enter student name & roll-no" << enal;
cin >> sname >> roll-no;
cout << " Enter marks 1 & marks2" << enal;
cin >> m1 >> m2;
per = ((m1 + m2)/2) * 100.
cout << "\n Percentage :" << per;
 }
};
int main() {
 marks m;
  m.cal();
  return 0;
}
```

O/P :

Enter Department name: *CSE*
Enter name & roll-no : 101
Enter marks 1 & marks 2 : 42
23
Percentage : 65 %.

b. Write a program to implement multiple. Assume suitable data.

→
```
#include <iostream>
using namespace std;
class student {
public:
 string name;
 void accept(){
  cout << "Enter student Name:";
  cin >> name;
 }
 void disp(){
 cout << "Name:" << name;
 }
};
class marks {
public:
int score;
void acept(){
cout << " Enter marks:";
cin >> score;
}
void disp(){
cout << " Marks:" << score;
};
class result : public student, public marks {
public:
 void disp(){
 student void accept(){
  acceptStudent();
  acceptMarks();
 }
 void disp(){
  dispStudent();
  dispStudent();
 }
};
int main(){
Result r;
r.accept();
 r.disp();
 return 0;
}
```

O/P

Enter student name: ABC
Enter marks: 34.

Name : abc
Marks : 34.

C. Write a program to implement hierarchial inheritance. Assume suitable data.

→
```
#include <iostream>
using namespace std;
class shape {
protected:
  int a,b;
public:
  void accept (){
  cout << "Enter two values:";
  cin>> a >> b;
  }
};
class rectangle :public shape {
public:
  void disp() {
  cout << "Rectangle area"<<(a*b)<<endl;
  }
};
class triangle : public shape {
public:
  void area (){
    accept ();
cout << "Area of triangle"<<(0.5 *a*b)<<endl;
}
};
```

```
int main() {
  Rectangle R;
  Triangle T;
  cout<<"\n Rectangle";
  R. area ();
  cout<<"\n Triangle";
  T. area ();
return 0;
}
```

O/P:-

Rectangle:
Enter two values: 5 4
Area of Rectangle = 20

Triangle:
Enter two values: 6,3
Area of triangle = 9

d). WAP to implement hybrid inheritance.

```
#include<iostream>
using namespace std;
class student {
public:
  int roll;
  void accept(){
    cout<<"Enter roll no:";
    cin>>roll;
  }
};
class test : public student {
public:
  int marks;
  void getmarks (){
  cout<<"Enter marks:";
  cin>>marks;
  }
};
class sports {
public:
  int score;
  void getscore (){
  cout<<"Enter sports score:";
  cin>>score;
  }
};
class result : public test , public sports {
public:
  void show(){
  cout<<"Roll no"<<roll<<endl;
  cout<<"Total"<<marks+score<<endl;
  }
};
int main() {
  result R;
  R.accept();
  R.getmarks();
  R.getscore();
  R.show();
  return 0;
}
```

O/P:-

```
Enter Roll no: 101
Enter marks: 75
Enter score: 200

Roll no: 101
Total: 275
```

e. WAP to virtual base class

```
#include <iostream>
using namespace std;
class student {
public:
  int roll;
  void getroll(){
  cout << "Enter roll :";
  cin >> roll;
  }
};
class test : virtual public student {
public:
  int marks;
void getmarks(){
cout << "Enter marks:";
cin >> marks;
  }
};
class sports : virtual public student {
public:
  int score;
  void getscore(){
cout << " Enter sports score:";
cin >> score;
};
class result : public test, public sports {
public:
    void show(){
    cout << "\n Roll no." << roll;
    cout << " \n Total" << marks + score << endl
    }
};
int main(){
 result R;
 R.getroll();
 R.getMarks();
 R.getscore();
 R.show();
 return 0;
}
```

O/P :-

```
Enter Roll no : 12
Enter marks : 11
enter sports score : 1

Roll no : 12
Total : 12
```

Experiment-7

a. WAP using function overloading to calculating area of laboratory (rectangle) & classroom (square).

```
#include <iostream>
using namespace std;
class area {
public:
  void calc(float len, float breadth){
  cout<<"Area of Laboratory :"<< len* breadth<<endl;
}
  void calc (float side){
    cout<< "Area of classroom:"<< side *side<<endl;
  }
};
int main (){
  area a;
  cout<<"Laboratory:"<<endl;
  a.calc(12.5, 10.0);
  cout << "classroom:"<<endl;
  a.calc (8.0);
  return 0;
}
```

b. WAP using function overloading to calculate sum of 5 float values & sum of 10 integer values.

```
#include <iostream>
using namespace std;
class sum {
public:
  float sum(float a, float b, float c, float d, float
  { return a+b+c+d+e; }
int main() sum (int arr[], int n){
  int s=0;
  for (int i=0; i<n; i++){
      s+=arr[i];
    return s;
  }
};
int main (){
  sum s;
cout<<"sum of 5 float no."<<s.sum (1.1,2.2,3.3,4.4,5.5
int arr[10] = {1,2,3,4,5,6,7,8,9,10};
cout<<"sum of 10 integers = "<<s.sum (arr,10);
return 0;
}
```

O/P:-

sum of 5 float no.s = 16.5

Sum of 10 integers = 55

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

→
```
#include<iostream>
using namespace std;
class number {
int num;
public:
 void setdata (int n) { num = n; }
 void display () { cout<<"Value = "<< num<<endl; }
 void operator -() {num = - num;}
};
int main() {
 number n;
 n.setdata (10);
 cout<< "Original :"; n.display ();
 -n;
 cout << "After Negation : "; n.display();
 return 0;
}
```

O/P-

Original : Value = 10
After Negation : Value = -10.

d. WAP to implement the Unary ++ operator (for pre & post increament)

→
```
#include<iostream>
using namespace std;
class counter {
int count;
public:
 void setdata (int c) { count = c; }
 void disp () { cout<< "Count ="<< count<<endl; }
 void operator ++(int) { count ++; }
};
int main () {
 counter c;
 c.setdata (5);
 cout << "After pre increment"; c.disp();
 c++;
 cout << "Original"; ; c.disp();
 c++;
 cout<<" After post increment :";
 return 0;
}
```

O/P

Original : count = 5
After Pre increment : count = 6
After Post increment : count = 7

Experiment-8:

a. Operator overloading

```
#include <iostream>
using namespace std;
class mystring {
  string str;
public:
 void setstring(){
 cout<< "Enter string: ";
  cin>> str;
 }
 void disp(){
 cout<< str<< endl;
 }
 mystring operator +(mystring s){
 mystring temp;
 temp.str = str + s.str;
  return temp;
  }
};
int main(){
mystring s1, s2, s3;
cout << "First string ";
s1.setstring();
cout << "second string ";
s2.setstring();
s3 = s1 + s2;
cout << "concatenated string: ";
s3.disp();
 return 0;
 }
```

O/P -

```
First string:
Enter string: abc
Second string:
Enter string: xyz
concatenated string : abc xyz
```

6. WAP to create a base class Ilogin having data members names & password. Declare & accept() function virtual. Derive Email login & Membershiplogin classes from Ilogin & Display login details of employee.

→
```
#include<iostream>
using namespace std;
class Ilogin {
protected:
    string name, password;
public:
  virtual void accept(){
  cout << "Enter name:";
  cin>> name;
  cout << "Enter password:";
  }
  virtual void disp() {
  cout << "Name:" << name << "Password:" << password;
  }
};
class EmailLogin : public Ilogin {
public:
  void accept() overide {
  cout<< "\n Email Login Details";
  Ilogin :: accept();
  }
  void display() override {
  cout << "Email Login ->";
  Ilogin :: display();
  }
};
class Membershiplogin : public Ilogin {
public:
  void accept() override {
  cout << "Membership login Details";
  Ilogin :: accept();
  }
  void display() override {
  cout<< "Membership login ->";
  Ilogin :: display();
  }
};
int main(){
  EmailLogin e;
  Membershiplogin m;
  e.accept();
  m.accept();
  cout<< "Displaying Details:";
  e.display();
  m.display();
```

return 0;
}

O/P:-

Email Login Details:
Enter Name: Alice
Enter Password: alice123

Membership login Details:
Enter name: Bob
Enter password: bob123

Displaying Login Details:
Email login → Name: Alice
password: alice123
Membership login → Name: Bob
password: bob123

Experiment-9

a. WAP to copy contents of one file into another.
Open "First.txt" in read (ios::in) mode & "Second.txt" file in write (ios::out) mode copy elements of first into second. Assume "First" is already created.

→
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
 ifstream fin("First.txt");
 ofstream fout("Second.txt");
 if(!fin) { cout<<"First.txt not found"; return 0; }
 string line;
 while (getline(fin, line)) {
  fout << line << endl;
 }
 fin.close();
 fout.close();
 cout<<"contents copied successfully";
 return 0;
}
```

O/P:-
First.txt not found

b. WAP to count digits & spaces using file handling

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
  ifstream fin ("First.txt");
  if (!fin) { cout << "File not found!\n"; return 0; }
char ch;
int digits = 0, spaces = 0;
while (fin.get(ch)) {
 if (isdigit(ch)) digits++;
 if (ch == ' ') spaces++;
}
fin.close();
cout << "Digits: " << digits << "Spaces" << spaces;
return 0;
```

O/P:-
File not found!

```
#include <iostream>
using namespace std;
class number {
 int num;
public:
 void setdata(int n) { num = n; }
 void disp() { cout << "Value = " << num << endl; }
 void operator -() { num = -num; }
};
int main() {
 number n;
 n.setdata(10);
 cout << "Original "; n.disp();
 -n;
 cout << "After negation: "; n.disp();
 return 0;
}
```

O/P:-
Original : Value = 10
After negation : Value = -10

c. WAP to count words using file handling.

```
#include<iostream>
#include<fstream>
#include<sstream>
using namespace std;
int main(){
ifstream fin("First.txt");
if(!fin){
cout<<"File not found";
return 0;
}
string word;
int count=0;
while(fin>>word){
count++;
}
fin.close();
cout<<"Total words: "<<count<<endl;
return 0;
}
```

O/P:-
File not found!

d. WAP a to count occurence of a given word using file Handling.

```
#include<iostream>
#include<fstream>
#include<sstream>
using namespace std;
int main(){
ifstream fin("First.txt");
if(!fin){cout<<"File not found!"; return 0;}
string word, target;
int count=0;
cout<<"Enter word to search";
cin>>target;
while fin>>word {
if(word==target) count++;
}
fin.close();
cout<<"The word "<<target<<"occurred"<<count
<<"times";
return 0;
}
```

O/P:-
File not found.

Experiment 10.

a. WAP to find sum of array elements using function template.

```
#include <iostream>
using namespace std;
template <class T>
T arrsum (T arr[], int n) {
T sum=0;
for (int i=0; i<n; i++)
    sum+ = arr[i];
  return sum;
}
int main(){
int Arr1 [5]= {1,2,3,4,5};
cout<<"sum of Array:"<< arrsum(Arr1,5);
}
```

O/P.

Sum of Array: 15.

b. WAP of square function using template specialization calculate square of integer no. & string. Write a specialized function for string.

→

```
# include <iostream>
# include <string>
using namespace std;
template <class T>
T sq(T n) {
  return n*n;
}
template <>
string sq <string> (string s){
 return s+s;
}
int main(){
int i=5;
string str="ABC";
cout<<"Square of Integer:"<< sq(i)<<endl;
cout<<"Square of string:"<< sq(str)<<endl;
}
```

O/P:-

Square of Integer: 25

Square of string: ABCABC

d) WAP to implement push & pop methods from stack using class template.

```
#include <iostream>
using namespace std;
template <class T>
class stack { T arr[5]; int top;
public:
stack() { top = -1; }
void push (T val) { if (top == 4) cout<< "Stack Over-
                    flow";
else
  arr[++top] = val;
}
void pop() { if (top == -1) cout << "Stack underflow";
else
cout << "Popped" << arr [top--] << endl; }
void disp () { cout << "Stack elements:";
 for(int i=0; i<= top; i++)
   cout << arr [i] << " ";
   cout << endl;
}
};
int main() {
stack <int> s;
s.push(10);
s. push (20);
s. push (30);
s. display());
s. pop();
s. display();
}
```

O/P

c. calculator. (10 operations)

```
#include <iostream>
using namespace std;
template < class T >
class calc { T num1, num2;
public:
  calc (T a, T b) {
   num1 = a;
   num2 = b;
  }
 void disp () {
 cout << "Addition : " << num1 + num2;
 cout << "Subtration : " << num1 - num2 ;
 cout << "Multiplication: " << num1 * num2;
 cout << "Division: " << num1 / num2 ;
 cout << "Modulus : " << num1 % num2;
 cout << "Square of num1 : " << num1 * num1 ;
 cout << "Square of num2 : " << num2 * num2;
 cout << "cube of num1 : " << num1 * num1 * num1 ;
 cout << "cube of num2 : " << num2 * num2 * num2;
 cout << "Average : " << (num1 + num2) / 2;
 cout << "Maximum : " << ((num1 > num2) ? num1
 : num2 );
 cout << "Minimum" << ((num1 < num2) ? num1 :
 num2 );
}
};
int main() {
 int a, b ;
 cout << " Enter two integers : ";
 cin >> a >> b;
 calc < int > calc (a, b);
 cout << " calculate : " << endl;
 calc . disp () ;
}
```

O/P :-

```
Enter two integers : 10 5
calculator
Addition : 15
Subtraction : 5
Multiplication : 50
Division : 2
Modulus : 0
Square of num1 : 100
Square of num2 : 25
cube of num1 : 1000
cube of num2 : 125
Average : 7
Maximum : 10
Minimum : 5
```

Experiment - 11

Q. Write a ~~vector~~ cpp program to implement generic vectors. Include following member functions :-
To create vector
a) To modify the value of given element
b) To multiply by a scalar value.
c) To display the vector in form (10, 20, 30 . . .)

→
```
#include <iostream>
#include <vector>
using namespace std;
template <typename T>
class vec {
  vector <T> v;
public:
  void create(int n) {
    v.resize(n);
    for (int i=0; i<n; i++) {
      cin>>v[i];
    }
  }
void modify(int index, T value) {
  if (index>=0 && index <v.size())
    v[index] = value;
  }
```

```
void multiply (T scalar) {
for (int i=0; i<v.size(); i++)
    v[i] *= scalar;
  }
void disp() {
cout << "C";
for (int i=0; i< v.size(); i++) {
 cout << v[i];
if (i != v.size()-1)
   cout << ", ";
 }
cout << ")" << endl;
  }
};
int main() {
vec < int > vec;
int n;
cin >> n;
vec.creat (n);
vec.disp ();
vec.modify (1, 50);
vec.disp();
vec.multipy(2);
vec.disp();
return 0;
}
```

Experiment-12.

Q. ~~Qu~~ WAP ~~to~~ using STL.

a) Implement stack.

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

int main() {
stack <int> S;
 s.push (10);
 s.push (20);
 s.push (30);
cout << "Top Element :" << s.top() << endl;
 s.pop();
cout << "Top after pop :" << s.top() << endl;
cout << "stack size:" << s.size () << endl;
return 0;
}
```

Output:

Top element : 30

Top after pop element : 20

stack size: 2

b. Implement Queue.

→
```
#include <iostream>
#include <queue>
using namespace std;
int main() {
  queue <int> q;
   q.push(10);
   q.push(20);
   q.push(30);
cout<<"Front : "<< q.front() << endl;
cout << " Back : " << q.back() << endl;
q.pop();
cout << "Front after pop : " << q.front() << endl;
cout << "Queue size : " << q.size() << endl;
return 0;
}
```

Output :-

Front : 10
Back : 30
Front after pop : 20
Queue size : 2.