1) Write a C program for addition of 2 numbers

```
#include <iostream.h>
using namespace std;
int main ()
{
   int a, b, c;
   cout << "enter value of a and b \n";
   cin >> a >> b;
   c=a+b;
   cout << "addition " << c;
   return 0;
}
```

Output

```
enter value of a and b
1 2
addition 3
```

2) Write a C++ program to perform arithmetic operations using switch case

```
#include <iostream>
using namespace std;
int main()
{
    int num1, num2, result;
    char op;
    cout << "enter first number";
    cin >> num1;

    cout << "enter an operator";
    cin >> op;

    cout << "enter second number";
    cin >> num2;

    switch (op)
    {
     case '+':
     result = num1 + num2;
     cout << "Result: " << result << endl;
     break;
     case '-':
       result = num1 - num2;
       cout << "Result: " << result << endl;
       break;
     case '*':
       result = num1 * num2;
       cout << "Result: " << result << endl;
```

```
        break;
    case '/':
      if(num1 != 0)
      {
        result = num1 / num2;
        cout << "Result:" << result << endl;
      }
      else
      {
        cout << "Division by 0 not allowed" << endl;
      }
      break;
    default:
      cout << "Invalid operator " << endl;
      break;
  }
  return 0;
}
```

Output-

enter first number: 2
enter operator: +
enter second number: 2
Result: 3

3) Write a C++ program to check if a number is even or odd.

```
#include <iostream>
using namespace;

int main()
{
  int number;
  cout << "enter number: ";
  cin >> number;
  if (number %.2 == 0)
  {
    cout << number << "is even " << endl;
  }
  else
  {
    cout << number << " is odd << endl;
  }
  return 0;
}
```

Output -

```
enter number : 3
3 is odd %.
```

4) Write a C++ program to print 1-10 numbers using for loop

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

int main()
{
  for(int i=1; i<=10; i++)
  {
    cout<<i<<" ";
  }
  cout<<endl;
  return 0;
}
```

Output.

1 2 3 4 5 6 7 8 9 10

5) Write a C++ program to print 10 - 1 using while loop

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

int main()
{
    int i=10;
    while (i>=1)
    {
        cout << i << " ";
        i--;
    }
    cout << endl;
    return 0;
}
```

Output

10 9 8 7 6 5 4 3 2 1

6) Write a C++ program to print the pattern

a)

```
    *
  *   *
*   *   *
```

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

int main()
{
  int rows = 3;
  int space;
  for(int i = 1; i <= rows; i++)
  {
    space = rows - i;
    for(int s = 1; s <= space; s++)
    {
      cout << " ";
    }
    for (int j = 1; j <= i; j++)
    {
      cout << "* ";
    }

    cout << endl;
  }
  return 0;
}
```

Output-

```
    *
  *   *
*   *   *
```

b) ~~#include <iostream>~~

```
1
2  2
3  3  3
```

```
#include <iostream>
using namespace std;
int main()
{
  int rows = 3;
  for (int i=1; i<=rows; i++)
  {
    for(int j=1; j<=i; j++)
    {
      cout<<i<<" ";
    }
    cout << endl;
  }
  return 0;
}
```

Output -

```
1
2  2
3  3  3
```

c)
```
#include<iostream>
using namespace std;
int main()
{
  int rows = 3;
  for(int i=1; i<=rows; i++)
  {
    for( int j=1; j<=i; j++)
    {
      cout<<j<<" ";
    }
    cout<<endl;
  }
  return 0;
}
```

1
1 2
1 2 3

Output-

```
1
1 2
1 2 3
```

8/7/25

EXPERIMENT-1

PROBLEM 1

Problem statement:

1) Write a C++ program to declare a class student having data members as roll no and name. Accept and display data for single student

CODE:

```
#include<iostream>
using namespace std;
class student
{
  int roll_no;
  string name;
  public:
  void accept()
  {
    cout<<" Enter student name and roll no: ";
    cin>> name >> roll_no;
  }
  void disp()
  {
    cout<<"Name of student : "<<name;
    cout<<" \n Roll no. : "<<roll_no;
  }
};
```

```
int main()
{
    student s1;
    s1.accept();
    s1.disp();
    return 0;
}
```

OUTPUT -

enter student name and roll no. abc 3
Name of student : abc
Roll no : 4

2) PROBLEM STATEMENT - write a C++ code to create a classbook having data members as Bname, Bprice, Bpages. Accept the data for 2 books and display the name of the book having greater price.

CODE:

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

class classbook
{
  public:
  int Bprice, Bpages;
  string Bname;
  public:
  void accept()
  {
    cout<<"Enter book name, price and pages";
    cin>> Bname >> Bprice >> Bpages;
  }
  void disp()
  {
    cout<<"The name of the most expensive book is "<<
    Bname;
  }
};
int main()
{
  classbook b1, b2;
```

```
    b1.accept();
    b2.accept();
    if(B1.Bprice > B2.Bprice)
    {
      b1.disp();
    }
    else
    {
      b2.disp()
    }
    return 0;
}
```

OUTPUT

Enter book Name, price and pages : abc 100 60
Enter book Name, price and pages : cde 50 100
The name of the most expensive book is abc.

3) PROBLEM STATEMENT - write a program to declare class "time", accept time in HH:mm:ss format, convert it to total seconds and display them.

CODE:

```
#include<iostream>
using namespace std;
class time
{
    public:
    int hour min sec;
    void accept()
    {
      cout<<"Enter time in hours, minutes and seconds
      cin >> " hour >> min >> sec;
    }
    void disp()
    {
      int s1 = (hour* 3600) + (min*60) + sec;
      cout << "The time in second " << s1;
    }
};
int main()
{
    class time t1;
    t1.accept();
    t1.disp();
}
```

15/7/25

OUTPUT:

Enter time in hours minutes, seconds: 1 2 3
The time in seconds is : 3723

EXPERIMENT - 2

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

Code:

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

class city
{
  public:
  string name;
  int population;
};
  void accept()
  {
  cout<<"Enter the name and population of 5 cities";
  cin>>name>>population;
  }
  void display()
  {
    cout<<"The name of the most populated city is";
  }
};
```

```
int main()
{
  city c[5];
  int i;
  for(i=0; i<5; i++)
  {
    c[i].accept();
  }
  for(i=0; i<5; i++)
  {
    c[i].display();
  }
  for(int i=0; i<5; i++
{
  int j;
  for(j=0; j<5; j++)
  {
    if((c[j].population > c[i].population)
    {
      break;
    }
  }
  if(j==5)
{
  cout<<"\n the most populated city is "<< c[
    << "with population" << c[i]population << end

}
return 0;
}
```

OUTPUT -

Enter the name and population of 5 cities

Delhi 100

Mumbai 200

Pune 300

Bangalore 400

Nagpur 50

The most populated city is bangalore

2 WAP to declare a class account having data member as account no. and balance. Accept this data for 10 account and give interest of 10%. where balance is equal or greater than 5000 and display them

Code:

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

class account
{
  public:
  int accountno;
  float balance
  void accept()
{
  cout<<"Enter account no.";
  cin >> account no;
  cout << "Enter balance";
  cin >> balance;
}
  void display()
{
  cout << "Acc no. "<<accountno << "Balance :"<<
      <<endl;
}
};
int main()
{
```

```
Account acc[10];
cout<<"enter detail for 10 accounts \n";
for(int i=0; i<10; i++)
{
 cout<<"\n Account"<<i+1<<": \n";
 acc[i].accept();
}
for(int i=0; i<10; i++)
{
  if(acc[i].balance>=5000)
  {
    acc[i].balance += acc[i].balance * 0.1F;
  }
}
cout<<"\n updated account detail: \n";
for(int i=0; i<10; i++)
{
   acc[i].display();
}
return 0;
}
```

OUTPUT -

Enter details for 10 account

1) Account no: 1
   Balance : 6000

2) Account no : 2
   Balance : 24000

3) Account no: 3
Balance : 65000

4) Account no: 4
Balance : 640006

5) Account no: 5
Balance : 5600

6) Account no : 6
Balance : 6500

7) Account no : 7
Balance : 5400

8) Account no : 8
Balance : 25600

9) Account no: 9
Balance : 40000

10) Account no : 10
Balance 10000

Updated account details

1 6600
2 26400
3 71500
4 75900
5 6160
6 7150

7 5940
8 2360
9 44000
10 11000

5) Write a program 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.

Code:

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

class staff
{
 public:
 string name;
 string post;
 void accept()
 {
  cout<<"enter staff name: ";
  cin>>name;
  cout<<"Enter staff post : ";
  cin>>post;
 }
};
int main()
{
 Staff staff array[5];
 cout<<" Enter details of 5 staff members :\n";
 for(i=0;i<n;i++)
 {
  cout<<"\n staff"<<i+1<<"\n ";
  staff Array[i].accept();
}
```

```
cout << "Enter details of 5 staff members : \n";
for(int i=0 ; i<5; i++)
{
  cout << "\n staff " << i+1 << "\n";
  staffArray[i].accept();
}
cout << "\n staff members " who are HOD: \n";
for(int i=0;i<5;i++)
{
  if(staffarray[i].post == "HOD")
  {
   cout<< staffarray[i].name<<endl;
  }
}
return 0;
}
```

Output:

Name : ABC
POST : HR

Name : ACB
POST : CEO

Name : BCD
Post : Clerk

Name : xyz
Post : HOD

29/7/25

Name : ZYX
Post : HOD

Staff member who are HOD : Ixyz
zyx

EXPERIMENT-3

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

CODE:

```
#include<iostream>
using namespace std;
class Book{
  public:
  string title;
  string author;
  float price;
  void accept()
  {
    cout<<"enter book title";
    cin>>title;
    cout<<"Enter author name";
    cin>>author;
    cout<<"Enter price";
    cin>>price;
  }
  void display()
  {
    cout<<"\n book title "<<title;
    cout<<"\n author name "<<author;
    cout<<"\n price "<<price;
  }
```

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

OUTPUT -

```
enter book title abc
enter author name e
enter price 2

book title abc
author name e
price 2
```

2) WAP to declare a class student having data members roll_no and percentage using 'this' pointer invoke member functions to accept and display this data for one object of the class.

CODE:

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

class Student
{
  int roll_no;
  float percentage
public:
  void accept()
  {
   cout<<"enter your roll no";
   cin>> roll_no;
   cout << "enter your percentage: ";
   cin>> percentage;
  }
  void display()
  {
   cout<< "Roll no: " << this -> roll_no <<endl;
   cout << "Percentage: " << this -> percentage <<endl;
  }
};
```

```
int main()
{
 Student S1;
 S1.accept();
 S1.display();
 return 0;
}
```

OUTPUT -

```
Enter your roll no : 4
Enter your percentage : 34
Roll no = 4
Percentage : 34
```

3) WAP to demonstrate the use of nested class

CODE:

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

Class student{
public:
  class marks{
      int marks;
    public:
      void accept()
      {
       cout<< "Enter the marks for the class:";
```

```
    cin >> marks;
  }
  void display() {
    cout<<"The marks are: "<<marks<<endl;
  }
  };
};
int main()
{
  student :: marks m1;
  m1.accept();
  m1.display();
  return 0;
}
```

OUTPUT-

```
Enter marks for the class: 5
The marks are: 5
```

5/8/25

Assignment:

1) Problem statement - WAP to swap two numbers by passing object as function argument by pass by value. Consider two no.s in same class

Code:

```
#include<iostream>
using namespace std;
class num
{
  int val;
  public:
  void accept()
  {
    cout<<"enter value of num1 ";
    cin>>val;
  }
  void disp()
  {
    cout<<val;
  }
  void swap(num n1, num n2)
  {
      int temp = n2.val;
      n1.val = n2.val;
      n2.val = temp;
      cout<<"num1 is: "<<n1.val<<endl;
      cout<<"num2 is: "<<n2.val<<endl;
  }
```

```
int main()
{
  num n1, n2, temp;
  n2.accept;
  n2.accept;
  temp.swap(n1, n2);
  return 0;
}
```

OUTPUT-

```
Enter value of num1 : 5
Enter value of num 2 : 6
num1 is : 6
num2 is : 5
```

2) Problem statement - WAP to swap two numbers by passing object as function argument by pass by value. Consider two numbers in different class

CODE:

```
#include<iostream>
using namespace std;
class B;
class A {
    int val1;
  public:
    void accept(){
      cout<< "enter value for first class: ";
      cin >> val1;
```

```
    }
    void disp(){
        cout << "value in in class A: " << val1 << endl;
    }
};
class B {
    int val2;
public:
    void accept(){
        cout << "enter value of second class: ";
        cin >> val2;
    }
    void disp(){
        cout << "value in B: " << val2; endl;
    }
    void swapvalues(A a, B b){
        int temp = a.val1;          <-- error
        a.val1 = b.val2;            <-- error
        b.val2 = temp;
        cout << "\n inside swap function: \n";
        cout << "value in A " << a.val1 << endl;
        cout << "value in B: " << b.val2 << endl;
    }
};
int main(){
    A obj1;
    B obj2, temp;
    obj1.accept();
    ob2.accept();
    cout << "Before swap : ";
    obj1.disp()
```

```
    obj2.disp();
    temp.swapvalues(obj1, obj2)
    obj1.disp();
    ob2.disp();
    return 0;
}
```

OUTPUT -

error: 'int A::val1' is private within this context

Problem statement:

3 WAP to add two complex no.s by passing object as function argument by pass by value.

Code:

```
#include<iostream>
using namespace std;
class c{
    int re, im;
    public:
    void get(){
        cout<<"\n enter real part: ";
        cin>> re;
        cout<<"\n enter imaginary part: ";
        cin>> im;
    }
    void disp(){
        cout<< re<<"+"<<im<<"i\n";
    }
```

```
    void sum(c c2, c c2) {;
        re = c1.re + c2.re;
        im = c1.im + c2.im;
    }
};
int main(){
    c c1, c2, c3;
    cout<<"enter 1st complex no. : \n";
    c1.get();
    cout<<" enter 2nd complex no: \n";
    c2.get()
    c3.sum(c1, c2;
    cout<<" \n the sum is " ;
    c3.disp();
    return 0;
}
```

OUTPUT -

```
enter 1st complex no:
enter real part: 4
enter imaginary part: 5
enter 2nd complex no. :
enter real part : 3
enter imaginary part: 2
the sum is : 7+7i
```

4) Problem statement: WAP to add two complex no.s by passing object as function argument by pass by reference.

Code:

```
#include <iostream>
using namespace std;
class c {
    int re, im;
    public:
    void get() {
    cout << " enter real part: ";
    cin >> re;
    cout << " enter imaginary part : ";
    cin >> im;
} }
    void disp () {
      cout << re << " + " << im << " i \n";
    }
    void sum ( c &c1 , c &c2 );
};
void c::sum ( c &c1, c &c2 ) {
    re = c1.re + c2.re;
    im = c1.im + c2.im;
}
int main () {
    c c1, c2, c3;
    cout << " enter First complex no. \n";
    c1.get ();
    cout << " enter second complex no. ";
    c2.get ();
    c3.sum (c1, c2);
```

cout<< "the sum is : ";
c3.disp();
return 0;
}

OUTPUT

enter first complex no.
enter real part : 5
enter imaginary part : 7
enter second complex :
enter real part : 6
enter imaginary : 5
the sum is : 11 + 12i

Page No.
Date

Friend function practice questions

1) Problem statement - Create two classes class A and class B, each with a private integers. Write a friend function sum() that can access private data from both the classes and return the sum.

Code:

```
#include<iostream>
using namespace std;
class B;
class A{
  int n1;
  public:
  void accept()
  {
    cout<<"enter number 1 : ";
    cin>>n1;
  }
  friend intsum(A, B);
};
class B{
  int n2;
  public:
  void accept()
  {
    cout<<"enter no.2";
    cin>>n2;
  }
  friend int sum(A, B);
};
```

```
int sum (A num1, B num2)
{
  return num1.n1 + num2.n2 ;
}
int main()
{
  A obj1;
  B obj2;
  obj1.accept();
  obj2.accept();
  cout << "sum is" << sum(obj1, obj2) << endl;
  return 0;
}
```

OUTPUT:

enter number 1: 3
enter number 2: 4
sum is 7

2) Problem statement - WAP with class number that contains a private integer. Use a friend function swapNumbers (Number&, Number&) to swap the private values of two number objects.

Code:

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

```
{
  int n;
  public:
  void accept()
  {
    cout<< "enter a number:";
    cin>> n;
  }
  void disp()
  {
    cout <<"\n num is : "<<n;
  }
  friend void swap (number &num1, number &num2);
};
void swap (number &num1, number &n2);
{
   int temp = num1.n;
   num1.n = num2.n;
   num2.n = temp;
}
int main()
{
   number n1;
   number n2;
   num 1.accept();
   num 2. accept();
   cout << " original values ar : \n";
   num1.disp();
   num2.disp();
   swap (num1, num2);
   cout << " \n swapped values are: \n";
```

```
    num1.disp();
    num2.disp();
    return 0;
}
```

OUTPUT -

```
enter number: 5
enter number: 4
original values are:
num is: 5
 num is: 4
swapped values are:
num is: 4
num is 5
```

3) Problem statement: Define two classes Box and cube, each having a private volume. Write a friend function find greater (Box, cube) that determines which object has a larger volume.

Code:

```
#include<iostream>
using namespace std;
class cube;
class box
{
   int volume;
   public:
```

```
    void accept()
    {
      cout << "enter volume of box: ";
      cin >> volume;
    }
    friend void findgreater(box vol1, cube vol2);
};
class cube
{
    int volume;
    public:
    void accept()
    {
      cout << "enter volume of cube:";
      cin >> volume;
    }
    friend void findgreater(box vol2, cube vol2);
};
void findgreater(box vol1, cube vol2)
{
    if(vol1.volume > vol2.volume)
    {
      cout << "enter volume of box is greater";
    }
    else
    {
      cout << " volume of cube is greater";
    }
}
int main()
```

```
{
  box vol1;
  cube vol2;
  vol1.accept();
  vol2.accept();
  findgreater(vol1, vol2);
  return 0;
}
```

OUTPUT:

```
enter volume of box : 56
enter volume of cube : 77
volume of cube is greater.
```

4) Problem statement - Create a class complex with real and imaginary parts as private. Use friend function to add two complex numbers and return the result as a new complex object

Code:

```
#include<iostream>
using namespace std;
class complex
{
  int real;
  int imaginary;
public:
  void accept()
  {
```

```
    cout<<"\n enter real part : ";
    cin>> real;
    cout<<"\n enter imaginary part: ";
    cin>>imaginary;
  }
  friend void add(complex n1, complex n2);
};
void add (complex n1, complex n2)
{
  int sum_real = n1.real+n2.real;
  int sum_imaginary = n1.imaginary + n2.imaginary;
  cout<<" Final complex no: "<<sum_real<<"+"
  <<sum_imaginary<<"i";
}
int main()
{
  complex n1, n2;
  n1.accept();
  n2.accept();
  add (n1, n2);
  return 0;
}
```

OUTPUT -

```
enter real part :6
enter imaginary : 8
enter real part : 3
enter imaginary : 6
final complex no : 9+14i
```

5) Problem statement - Create a class student with private members: Name and three subject marks. Write friend function calculate Average (student) that calculates and displays the averages marks.

Code:

```
#include<iostream>
using namespace std;
class student
{
  string name;
  int m1, m2, m3;
  public:
  void accept()
  {
    cout<<"\n enter name of student:";
    cin>>name;
    cout<<" enter marks of first subject :";
    cin>>m1;
    cout<<"enter marks of second subject :";
    cin>>m2;
    cout<<"enter marks of third subject :";
    cin>>m3;
  }
};
void average(student s1)
{
  int avg=(s1.m1+s1.m2+s1.m3)/3;
  cout<<"average is :"<<avg;
```

```
}
int main()
{
  student S1;
  S1.accept();
  average(S1);
}
```

OUTPUT -

```
enter name of student : shreyash
enter marks of first subject : 76
enter marks of second subject : 68
enter marks of third subject : 45
average : 69
```

6) Problem statement - Create three classes Alpha, beta, gamma, each with a private data member. Write a single friend function that can access all 3 and print their sum.

Code:

```
#include<iostream>
using namespace std;
class Beta;
class Gamma;
class alpha
{
  int n;
```

```
    public:
    void accept()
    {
      cout<<"enter value of alpha ";
      cin >>n;
    }
    friend void sum(alpha A, Beta B, Gamma C);
};
class Beta
{
  int n;
  public:
  void accept()
  {
    cout<<" \n enter value for Beta : ";
    cin>>n;
  }
  friend void sum(alpha A, Beta B, Gamma C);
};
class Gamma
{
  int n;
  public:
  void accept()
  {
    cout<<" \n enter value for gamma : ";
    cin>>n;
  }
  friend void sum(alpha A, Beta B, Gamma C);
};
```

```
void sum(alphaA ,Beta B, Gamma C)
{
  int s=A.n+B.n+C.n;
  cout<<" the sum is: "<<s;
}
int main()
{
  alpha A;
  Beta B;
  Gamma C;
  A.accept();
  B.accept();
  C.accept();
  sum(A,B,C);
  return 0;
}
```

OUTPUT -

```
enter value for alpha: 5
enter value for Beta: 6
enter value for Gamma: 7
the sum is: 18
```

7) Problem statement - Create a class point with private members x and y. Write a function that calculates and returns the distance between two point objects

Code:

```
#include<iostream>
#include<cmath>
using namespace std;
class point{
  int x, y;
  public:
  void accept()
  {
    cout<<"\n enter x value: ";
    cin>>x;
    cout<<"\n enter y value: ";
    cin>>y;
  }
  friend void calculate(point P1, point P2);
};
void calculate(point P1, point P2);
{
  double distance=sqrt(P2.x-P1.x)*(P2.x-P1.x)+
  (P2.y-P1.y)*(P2.y-P1.y));
  cout<<"The distance between the two points is: "<<d
}
int main()
{
  point P1,P2;
  P1.accept();
  P2.accept();
  calculate(P1, P2);
  return 0;
}
```

OUTPUT-
enter x value: 5
enter y value: 6
enter x value: 3
enter y value : 7
the distance between two points: 2.23607

6) Problem Statement: Create two classes : Bank account and Audit. Bank Account holds private balance information. Write a friend function in Audit that accesses and prints balance information for audit.

Code:

```
#include<iostream>
using namespace std;
class audit;
class bankaccount
{
  string name;
  int bank balance;
  public:
  void accept()
  {
    cout<<"\n enter: ";
    cin>>name;
    cout<<"\n enter bank balance: ";
    cin>>bankbalance;
  }
  friend audit;
};
```

```
class audit
{
  public:
  void startAudit( bank account b1 )
  {
    cout<< "\n name: " << b1.name;
    cout<<"\n bank balance : " << bank balance;
  }
};
int main()
{
  bank account b1;
  audit A1;
  b1.accept();
  A1.startaudit(b1);
  return 0;
}
```

OUTPUT-

```
enter name: Shreyash
enter balance: 1234
name: Shreyash
bank balance: 1234
```

1218

EXPERIMENT-4

1) Problem statement - WAP to swap two numbers from same class using function argument. Write swap function as member function

Code:

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

class num{
public:
  int n1, n2;
  void accept(){
  cout<<" enter first no. "<<endl;
  cin>>n1;
  cout<<"enter second no. "<<endl;
  cin>>n2;
  }
  void display(){
    cout<<"Num1 : "<<n1<<endl;
    cout<<"Num 2: "<<n2<<endl;
    }
    void swap (num&x) {
      int temp = x.n1;
      x.n1 =x.n2;
      x.n2 = temp;
      cout<<"swapped first no."<<x.n1<<" second no.
      <<x.n2<<endl;
    }
```

```
    }
};
int main(){
    num nu;
    nu.accept();
    nu.display();
    nu.swap(nu);
    nu.display();
    return 0;
}
```

OUTPUT-

```
enter first no: 9
enter second no: 7
num1: 9
num2: 7
Num1: 7
num2: 9
```

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

Code:

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

class num {
    int n1, n2;
```

```
public:
void accept(){
cout << "enter first no.:";
cin >> n1;
cout << "enter second no:";
cin >> n2;
}
void display(){
cout << "num 1:" << n1 << endl;
cout << "num 2:" << n2 << endl;
}
friend void swap numbers (num &obj);
};
void swapnumbers (num &obj){
int temp = obj.n1;
obj.n1 = obj.n2;
obj.n2 = temp;
cout << "Numbers swapped successfully";
}
int main(){
num obj;
obj.accept();
obj.display();
swapnumbers (obj);
obj.display();
return 0;
}
```

OUTPUT-

Enter first no: 6
enter second no: 5
Num1: 6
Num2: 5
Num1: 5
Num2: 6
Numbers swapped successfully.

9) Problem statement: WAP to swap two no.s from different class using friend function

Code:

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

class num1{
public:
    int n1;
    void accept()
    {
      cout<<"enter first no: "<<endl;
      cin>>n1;
    }
    void display()
    {
      cout<<"Num1:"<<n1;
    }
  friend void swapnumbers(num1 &a, num2 &b);
};
```

```
class num 2{
public: int n2;
    public:
public:
;
void accept(){
  cout<<"enter second no: ";
  cin >>n2;
  }
void display (){
  cout<<"Num 2: "<<n2<<endl;
  }
  friend void swapnumbers(num1 &a , num1 b);
};
Void swapnumbers (num n1&a, num& b){
    int temp = a.n1;
    a.n1 =b.n2;
    b.n2 = temp;
    cout<<"no.s swapped"<<endl;
}
int main(){
  num1 obj1;
  num2 obj2;
  obj1.accept();
  obj2.accept();
  cout<<"before swap"<<endl;
  obj1.display();
  obj 2.display();
  swapnumbers (Obj1, obj2);
  cout<<"after swap";
```

```
    obj1.display();
    obj2.display();
    return 0;
}
```

OUTPUT -

```
Enter first no: 5
Enter second no: 6
No.s swapped
Befor swapping:
 num1: 5
 num2: 6
Afters wapping
 num1: 6
 num2: 5
```

4) WAP to create two classes Result1 and Result2 which stores the marks of students. Read the value of marks for both the class objects and compute the average of two results.

Code:

```
#include<iostream>
using namespace std;
class result2;
class result1{
  int r1;
 public:
```

```
void accept()
{
cout << "enter first result : ";
cin >> r1;
}
friend float avg (Result1, Result2);
};
class result2 {
int r2;
public :
void accept(){
cout<<"Enter second result : ";
cin >> r2;
}
friend float avg(Result1, Result2);
};
float avg( Result res1, result res2){
return(res1.r1 + res2.r2) / 2.0f;
}
int main(){
Result1 res1;
Result2 res2;
res1.accept();
res2.accept();
float average = avg(res1, res2);
cout << "Average is: " << average << endl;
return 0;
}
```

OUTPUT -

Enter first result : 6
Enter second result: 5
Average is 5.5

5 Problem statement: WAP to find greatest no. among 2 no.s from two different classes using friend function

Code:

```
#include<iostream>
using namespace std;
class Number2;
class Number1{
 int n1;
 public:
 void accept()
 {
   cout<<"Enter First no.";
   cin>>n1;
  }
   friend void greatest(number1, number2);
};
class Number2
 int n2;
 public:
  void accept (){
   cout<<"enter second no:";
```

```
    cin >> n2;
  }
  friend void greatest(Number1, Number2);
};
void greatest(Number1 num1, Number2 num2){
  if numbers(num1.n1 > num2.n2){
    cout<<" greatest is :"<< num1.n1 <<endl;
  }else if (num2.n2 > num1.n1){
    cout <<" The greatest no. "<< num2.n2<<endl;
  }else{
    cout <<" Both no.s are equal "<<endl;
  }
}
int main(){
  Number1 num1;
  Number2 num2;
  num1.accept();
  num2.accept();
  greatest(num1, num2);
  return 0;
}
```

OUTPUT -

Enter first no: 5

Enter second no: 6

The greatest no is 6

1/18/25

EXPERIMENT - 5

1) Problem statement-

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 using

(i) Default constructor:

code:

```
#include<iostream>
using namespace std;
class sum
{
  int n, result;
  public:
  sum()
  {
    cout<<"\n enter value of number:";
    cin>>n;
    for(int i=0; i<=n; i++)
    {
      result=result+i;
    }
    cout<<"\n the sum is :"<<result;
  }
};
```

```
int main()
{
  sum obj;
  return 0;
}
```

OUTPUT:

```
enter value of number : 6
the sum is : 21
```

(ii) Parameterized consmuctor

Code:

```
#include<iostream>
using namespace std;
class sum{
  int n, result;
public:
  Sum(int sum){
    result = 0;
    for(int i=1; i<=num; i++)
    {
      result += i;
    }
    cout<<"Sum is :"<<result;
  }
};
```

```
int main(){
  int n;
  cout<<"enter value of n: ";
  cin>>n;
  Sum obj(n);
  return 0;
}
```

(iii) Copy constructor

```
#include <iostream>
using namespace std;
class Sum {
  int n, result;
  public:
  Sum(){
    n = 5;
    result = 0;
  }
  sum(const Sum &obj){
     n=obj.n;
     result = 0;
     for(int i=1; i<=n; i++){
       result += i;
     }
     cout<<"Sum is: "<<result<<endl;
  }
};
int main (){
  Sum s1;
```

Sum s2(s1);
return 0;
}

OUTPUT -

Sum is : 10

2) Problem statement:

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

(i) Default constructor:

Code:

```
#include<iostream>
using namespace std;
class student {
  string name;
  float percentage;
public:
  Student(){
    name = "Shreyash";
    percentage = 99;
  }
  void display(){
    cout<<"name: "<<name;
    cout<<"\n percentage: "<<percentage;
  }
};
int main(){
  student S;
  S.display()
  return 0;
}
```

OUTPUT :
name: Shreyash
percentage : 99

ii) Parameterized constructor

```
#include<iostream>
using namespace std;
class student
{
  float per;
  string name;
  public:
  Student(float p, string n)
  {
    per = p;
    name = n;
  }
  void disp()
  {
    cout<<"Percentage:"<<per;
    cout<<"name:"<<name;
  }
};
int main()
{
  Student S1(99, "Shreyash");
  S1.disp();
  return 0;
}
```

OUTPUT:

percentage = 99
name = Shreyash

(iii) Copy constructor

```
#include <iostream>
using namespace std;
class Student {
    String name;
    float percentage;
    public :
    Student () {
      name = "Shreyash" ;
      percentage = 99 ;
    }
    Student (const Student &obj)
    {
      name = obj.name;
      percentage = obj.percentage;
      cout << " copy constructor Called";
    }
    void display () {
    cout << "name: " << name;
    cout << \n percentage : " << percentage ;
    }
};
```

int main () {
Student S;
S.display ();
Student S2(S);
S2.display ();
return 0;
}

OUTPUT:

name: shreyash
percentage: 99
copy constructor called
name: Shreyash
percentage: 99

Problem statement: Define a class "college" members variables as roll_no, name, course. WAP using constructor with default value as Computer engineering for this course. Accept data for two Objects of class and display data

Code:

```
#include <iostream>
using namespace std;
class college
{
  int roll;
```

```
    string name, course;
    public:
    college (int roll, string name, string course = "Comp. Eng")
    {
      int r=roll;
      string n=name;
      string c=course;
      cout<<r<<n<<c;
    }
};
int main()
{
  college c1(1, "Shreyash");
  return 0;
}
```

OUTPUT-

1 Shreyash Comp Eng

4) Problem statement: WAP to demonstrate constructor overloading

Code:

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

```
int l, b;
public:
rectangle()
{
  l = 1;
  b = 2;
}
rectangle(int x)
{
  l = x;
  b = x;
}
rectangle(int x, int y)
{
  l = x;
  b = y;
}
void area()
{
cout << "\n area is " << l * b;
}
};
int main()
{
rectangle r;
r.area();
rectangle r2(4);
r2.area();
rectangle r3(1, 2);
r3.area();
return 0;
}
```

OUTPUT -

area is 2
area is 16
area is 2

(2)

6/9

EXPERIMENT - 6

1) Single inheritance
Problem statement: Create a base class called person with attributes name and age. Derive a class student from person that adds an attribute roll no. Write functions to display all details of student

Code:

```
#include<iostream>
using namespace std;
class person
{
  protected:
  string name = "Shreyash";
  int age = 17;
};
class student : protected person
{
  int roll=21;
  public:
  void display()
```

```
{ cout << "Roll no is : " << roll << "\n name is : " << name
<< "\n age is:" << age;
}
};
int main()
{
Student S1;
S1.display();
return 0;
}
```

OUTPUT:

```
roll no is: 21
name is : Shreyash
age is : 17
```

2) Multiple inheritance

Problem statement: Create two base classes academics and Sports. Write a function to calculate the roll, score and display details

Code:

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

```
protected:
int marks = 80;
};
class sports
{
protected:
int sportsscore = 50;
};
class result : protected academics, protected sports
{
int totalscore = marks + sportsscore;
public:
void calculate()
{
cout<< "total score is: " << totalscore
}
};
int main()
{
result r1;
r1.calculate
```

output

total score is : 130

3) multilevel inheritance

Problem statement: Create a class vehicle with attributes like brand and model. Derive a class car from vehicle which adds an attribute type

(eg. sedan (SUV). Further derive a class electric car from car which adds battery capacity. Write functions to display all the details

Code:

```
#include<iostream>
using namespace std;
class vehicle
{
  protected:
  String brand = "bwww";
  String model = "x1";
};
class car: protected vehicle
{
  protected:
  String type = "SUV";
};
class electriccar : protected car
{
int battery capacity = 80;
public :
 void display()
{
  cout<< "\n brand is : "<< brand;
  cout << "\n model is: "<< model;
  cout<< "\n type is: "<< type;
  cout <<"\n battery capacity is : "<< batterycapacity;
}
};
```

```
int main()
{
  electricar e1;
  e1.display();
  return 0;
}
```

O/P:
brand is : BMW
model is : x1
type is : SUV
battery capacity is : 80

4 Hierarchical inheritance

Problem statement: Create a base class employee with attributes empID and name. Derive two classes manager and developer from employee. Manager has an attribute department and developer has an attribute programming language. Write functions to display details of both.

Code:

```
#include<iostream>
using namespace std;
class employee
{
  protected:
  int empID = 5;
  string name = "Shreyash";
};
class manager: protected employee
{
  string dept = "poly";
```

```
public:
void display()
{
cout<<"\n empid : "<<empID;
cout<<"\n name is :"<<name;
cout<<"\n dept name is : "<<dept;
}
};
class developer : protected employee
{
string programminglanguage = "CPP";
public:
void display()
{
cout<<"\n empID is : "<<empID;
cout<<"\n name is : "<<name;
cout<<"\n programming language : "<<programmingla
nguage;
}
};
int main()
{
manager m1;
developer d1;
m1.display();
d1.display();
return 0;
}
```

OUTPUT:

empid is 5
name is Shreyash
department name is poly
emp id is 5
name is shreyash
programming language is cpp.

5 Hybrid inheritance

Problem Statement: Combine multilevel and multiple inheritance. Create a base class person with attributes name and age. Derive class student from person. Create two classes sports and academic. Derive class result from student and sports.

Code:

```
#include<iostream>
using namespace std;
class person
{
   protected:
   string name = "Shreyash";
   int age=17;
};
class student: protected person
{
   protected:
```

```
    int academicsmarks = 90;
};
class sports
{
  protected:
  int sportsscore = 50;
};
class academics
{
  protected:
  string grade = "A";
};
class result : protected student, protected sports, protected
academics
{
  int totalscore = sportsscore + academics marks;
  public:
  void calculate()
  {
    cout<< "Totalscore is : " << totalscore << "\n
    grade is : " << grade;
  }
};
int main()
{
  result r1;
  r1.calculate();
  return 0;
}
```

OUTPUT :

totalscore is : 140
grade is : A

6 Virtual base class)

Problem statement: WAP to implement following inheritance (use virtual base class)

class : college student
data mem student ; id

class : test
data mem : percentage

class : sports
data mem : grade

class : result

Code:

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

class collegestudent {
protected :
    int student_id ;
    int college_code ;
public :
```

```
void collectdata(){
  cout<<"enter student ID : ";
  cin >> student_id;
  cout<<"enter college code: ";
  cin >> college_code;
}
void display(){
  cout<<"\n student id:"<<student_id;
  cout<<"\n college_code: "<<college_code;
}
};
class Test : virtual public collegestudent{
protected:
  float percentage;
public:
  void gettestdata(){
    cout<<"enter percentage: ";
    cin >> percentage;
  }
  void displaytestdata(){
    cout<<"\n percentage :"<<percentage;
  }
};
class sports : virtual public collegestudent{
protected:
  char grade;
public:
  void getsportsdata(){
    cout<<"enter sports grade: ";
    cin >> grade;
  }
```

```
    void displaysports data () {
      cout << "\n sports grade : " << grade;
    }
};

class Result : public test, public sports {
public :
    void getdata () {
      collect data ();
      get test data ();
      get sports data ();
    }
    void display data () {
      cout << " result : \n";
      display ();
      display test data ();
      display sports data ();
    }
};
int main () {
  Result r;
  r.getdata ();
  r. display data ();
  return 0;
}
```

OUTPUT:

enter student ID : 1
enter college code : 34

enter percentage: 88
enter sports grade: A

result:
student ID: 1
college code: 34
percentage: 88
sports grade: A

23/9/25

EXPERIMENT - 7

1) Problem statement: WAP using function overloading to calculate the area of a laboratory (which is rectangular in shape) and area of classroom (which is square in shape)

Code:

```
#include <iostream>
using namespace std;
class calculate area
{
    public:
    void area(int l, int b)
    {
      cout << "\n area of lab is: " << l*b;
    }
    void area (int s)
    {
```

```
        cout << "\n area of class is " << s*s;
    }
};
int main()
{
    Calculate area A;
    int l, b, side;
    cout << " enter l, b, side \n";
    cin >> l >> b >> side;
    A.area (l, b);
    A.area(side);
    return 0;
}
```

OUTPUT:

```
enter l, b, side : 1 2 3
area of lab is : 2
area of class is : 9
```

2) Problem Statement: Write a program using function overloading to calculate sum of 5 float values and sum of 10 integers.

Code:

```
#include<iostream>
using namespace std;
class sumover{
    int num, sum = 0;
    float fnum, fsum = 0.0;
```

```
public:
void sumint(int){
  for(int i=0; i<10; i++)
  {
     cout<<"enter the integer";
     cin >> num;
     sum =sum+num;
  }
     cout<<sum;
}
void sumint(int){
  for (int i=0; i<10; i++)
  {
     cout<<"enter the integer ";
     cin >> num;
     sum = sum+num;
  }
  cout<< sum << "\n";
}
void sumint(float){
  for(int i=0; i<5; i++)
  {
     cout<<" enter the float";
     cin >> fnum;
     fsum = fsum + fsum;
  }
  cout<<fsum;
}
};
```

```
int main() {
  sumover s;
  s.sumint(0);
  s.sumint(0.0f);
  return 0;
}
```

OUTPUT:

enter integer : 1
enter integer: 2
enter integer: 3
enter integer : 4
enter integer: 5
enter integer: 6
enter integer : 7
enter integer : 8
enter integer: 9
enter integer : 9
55
enter float: 3.3
enter float : 4.4
enter float : 4.4
enter float : 4.4
enter float: 2.1
18.6.

3) Problem statement - Write a program to implement unary operator when used with the object so that the numeric data member of the class is negated.

Code:

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

class C1
{
public:
  int A;
  void operator -()
  {
    A--;
  }
};
int main()
{
  C1 B;
  B.A = 25;
  cout << "Before : " << B.A << endl;
  -B;
  cout << "After : " << B.A << endl;
  return 0;
}
```

OUTPUT:

Before: 25

After 24

4) Problem statement unary ++ operator (for pre increment and post increment) when used with the object so that the numeric data member of the class is incremented

code:

```
#include<iostream>
using namespace std;
class addition
{
 public:
 int A;
 void operator ++()
 {
   A++;
 }
};
int main()
{
 addition B;
  B.A=25;
  cout<<"Before: "<<B.A<<endl;
  ++B;
  cout<<"After: "<<B.A<<endl;
  return 0;
}
```

OUTPUT:

Before: 25

After : 26

17/10

EXPEREMENT 8

1) Write a program to overload the '+' operator so that two strings can be concatenated.

Code:

```
#include <iostream>
#include <cstring>
using namespace std;
class string{
private:
  char str[50];
public:
  void accept(){
    cout<<"enter a string: ";
    cin>>str;
  }
  void display(){
    cout<<str<<endl;
  }
  void operator +(string s2){
    strcat(this->str,s2.str);
  }
};
int main(){
  string s1,s2;
  s1.accept();
  s2.accept();
  s1+s2;
  cout<<"After concat";
```

```
s1.display();
cout<<" value of s2 is: ";
s2.display();
}
```

OUTPUT:

```
Enter a string: k
Enter a string: o
After concat
ko
value of s2 is
o
```

2) Problem statement: write a program to create a base class ILogin having datamembers name and password. Declare accept() function virtual. Derive email login and membership login classes from

Code:

```
#include<iostream>
using namespace std;
class ILogin {
protected:
  string name, password;
public:
  virtual void accept(){
    cout<<"enter name"; and
    cin>>name;
```

```
cout<<"enter password: ";
cin>>password;
}
virtual void display(){
cout<<"name"<<name<<" password "<<password;
}
};
class emaillogin:public Ilogin{
string email;
public:
void accept(){
Ilogin::accept();
cout<<"enter email: ";
cin>>email;
}
virtual void display(){
cout<<"name:"<<name<<"Password: "<<password
<<"email:"<<email;
}
};
class membershiplogin:public Ilogin{
string memberID;
public:
void accept(){
Ilogin::accept();
cout<<"enter membership id";
cin>>memberid;
}
void display(){
cout<<"membership login \n name: "<<name<<"password"
```

```
    cc password cc "member id" cc membership;
  }
};
int main(){
  Ilogin *ptr;
  emaillogin e;
  membershiplogin m;
  ptr = &e;
  ptr -> accept();
  ptr= &m;
  ptr -> accept();
  ptr=&e;
  ptr -> display();
  ptr&m;
  ptr -> display();
  return 0;
}
```

OUTPUT -

Enter name : abc
Enter password : www
enter email : abc@gmail
enter name : ss
enter password : a
enter membership id : 4

Email Login:
Name: abc
password : ww
email : abc@gmail

Membership login:
Name: ss
password : a
member ID : 4

17/10

EXPERIMENT - 9

1) Copy content from one file to another

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  fstream first_file, second_file;
  first_file.open("first.text", ios::in);
  if(!first_file){
    cout<<"error opening";
    return 1;
  }
  second_file.open("second.txt", ios::out);
  if(!second_file){
    cout<<"error opening"
    return 1;
  }
```

```
    char ch;
    while (first_file.get(ch)){
        second_file.put(ch);
    }
    first_file.close();
    second_file.close();
    cout << "file copied successfully";
    return 0;
}
```

OUTPUT:

File contents copied successfully

2) Count digits and spaces

```
#include<iostream>
#include<fstream>
#include<ctype.h>
using namespace std;
int main()
{
    fstream new_file;
    new_file.open("first.txt", ios::in);
    if(!new_file){
        cout<<"error opening";
        return 1;
    }
```

```
int dig_c=0;
int spc_c=0;
char ch;
while (new_file.get(ch)){
  if(isdigit(ch)){
    dig_c++;
  }
  else if (isspace(ch)){
    spc_c++;
  }
}
new_file.close();
cout<<"no. of digits"<<dig_c<<"no. of spaces"<<
  spc_c;
  return 0;
}
```

OUTPUT -
No. of digits: 7
No. of spaces : 3

3) Count words

```
#include<iostream>
#include<fstream>
#include<ctype.h>
using namespace std;
int main()
{
  fstream new_file;
```

```
new-file.open("first.txt", ios::in);
if(!new-file){
 cout<<"error";
 return 1;
}
char ch;
int word_c=0;
bool inword=false;
while(new-file.get(ch){
 if (isspace(ch)){
  inword=false;
 }
 else if (!inword){
   word_c++
   inword=true;
 }
}
new-file.close();
cout<<"no. of words "<<word_c;
return 0;
}
```

OUTPUT:
NO. of words: 5

4) Find occurence of a given word

Code:

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
int main()
{
fstream new_file;
new_file.open("first.txt", ios::in);
if(!new_file){
cout<<"error";
return 1;
}
string target = "hello";
string word;
int count=0;
while(new_file>>word){
if(word==target){
count++;
}
}
new_file.close();
cout<<"The word"<<target<<"occured"<<count
<<"times";
return 0;
}
```

OUTPUT

The word hello occured 3 times

EXPERIMENT - 10

1) Sum using function template

Code:

```
#include<iostream>
using namespace std;
template<typename T>
T sumArray(T arr[]){
    T sum=0;
    for(int i=0; i<10; i++){
        sum+=arr[i];
    }
    return sum;
}
int main(){
    int intArr[10]={1,2,3,4,5,6,7,8,9,10};
    float floatArr[10]={1.1f,2.2f,3.3f,4.4f,5.5f,
    6.6f,7.7f,8.8f,9.9f,10.0f};
    int intSum=SumArray(intArr);
    cout<<"sum of integer array: "<<intSum<<endl;
    float floatsum=sumarray(floatarr);
    cout<<"sum of floatarray "<<floatSum;
    return 0;
}
```

OUTPUT:

sum of integer array: 55
sum of floatarray 59.5

2) Square using function template

Code:

```
#include<iostream>
#include<string.h>
using namespace std;
template<typename T>
T square(T val)
{
  T sq = val*val;
  cout<<sq;
  return sq;
}
template<>
string square(string sqvalue)
{
  string squaree = sqvalue+sqvalue;
  cout<<squaree;
  return squaree;
}
int main()
{
  int A=5;
  string B="abc";
  square(A);
  square(B);
  return 0;
}
```

OUTPUT:

25abcabc

3) Simple calculator using class template

code:

```
#include<iostream>
using namespace std;
template<class T>
class C
{
  T num1, num2;
public:
  C(){num1=0; num2=0;}
  void accept()
  {
    cout<<"enter numbers:";
    cin>>num1>>num2;
  }
  void add()
  {
    cout<<"sum:"<<num1+num2;
  }
  void sub()
  {
    cout<<"sub"<<num1-num2;
  }
  void mul()
  {
    cout<<"mul"<<num1*num2;
  }
  void div()
  {
```

```
if(num2!=0)
{
  cout<<"div "<<num1/num2;
}
else
{
  cout<<"dont divide by 0";
}
};
int main()
{
  Calc obj;
  obj.accept();
  obj.add();
  obj.sub();
  obj.mul();
  obj.div();
  return 0;
}
```

OUTPUT:

Enter numbers: 5

6

sum : 11

Sub: -1

Mult: 30

Div: 0

4) Push and pop using class template

```
#include<iostream>
using namespace std;
template<class T>
class Stack{
  T stack[100];
  int top;
public:
  Stack(){
    top=-1;
  }
  void push(T value){
    if(top=99){
      cout<<"overflow";
    }else
    {
      top++;
      stack[top]=value;
      cout<<value<<'ipushed into stack";
    }
  }
  void pop(){
    if(top==-1){
      cout<<"underflow";
    }else
    {
      cout<<stack[top]<<"popped from stack";
    }
  }
```

```
void display (){
if(top == -1){
  cout << "Stack is empty";
}else
{
   cout << "Stack is empty";
}else{
   cout << "Stack elements";
   for(int i=top; i>=0; i--)
   {
      cout<<stack[i]<<" ";
   }
  }
};
int main(){
  Stack<int> s;
  s.push(10);
  s.push(20);
  s.push(30);
  s.display()
  s.pop()
  s.display();
  return 0;
}
```

OUTPUT:

10 pushed into stack
20 pushed into stack
stack elements : 20 10
20 popped from stack

EXPERIMENT - 11

1) Accessing vector using iterators

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
int main()
{
  vector<char> v(10);
  vector<char>::iterator p;
  int i;
  p=v.begin();
  i=0;
  while(p!=v.end()){
      *p=i+a;
       p++;
       i++;}
  cout<<"Original content:\n";
  p=v.begin();
  while(p!=v.end()){
     cout<<*p<<" ";
     p++;
};}
```

O/P:

vector contents

a b c d e f g h i j

2) Without iterator:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
int main()
{
 vector<char> v(10);
 unsigned int i;
 cout<<"size ="<<v.size()<<endl;
for(i=0; i<10; i++)
{
  v[i] = i+a;
}
cout<<"current content:";
for(i=0; i<v.size(); i++){
 cout<<v[i]<<" ";
}
cout<<"\n\n";
cout<<"expanding vector:";
for(i=0; i<10; i++)
{
 v.push_back(i+10+'a');
}
cout<<"size now="<<v.size()<<endl;
cout<<"current contents:\n";
for(i=0;i<v.size();i++)
{
  v[i]=toupper(v[i]);
}
```

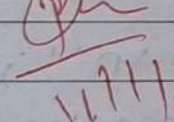
```
cout<<"modified contents \n";
for(i=0;i<v.size();i++)
{
  cout<<v[i]<<" ";
}
return 0;
}
```

O/P:

Current elements in the vector are:
a b c d e f g h i j

Expanding vector:

A B C D E F G H I J



Exp-12

a) Implement stack

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
  stack<char> cars;
  cars.push("BMW");
  cars.push("Audi");
  cars.push("Meracles");
  cars.push("Ferrar");
  cout<<"Top element is :"<<cars.top()<<endl;
  cout<<"size of stack is :"<<cars.size()<<endl;
  cars.pop();
  cars.pop();
  while(!cars.emphy())
  {
    cout<<"elements in stack are:"<<cars.top();
    cars.pop();
  }
  return 0;
}
```

o/p:

Top element : Ferrari
size of stack is : 4
Element stack are: Audi
" " : BMW

b) Implement queue

```
#Include<bits/stdc++.h>
using namespace std;
int main(){
queue<int> age;
age.push(21);
age.push(22);
age.push(23);
age.push(24);
cout<<"Front element is: "<<age.front();
cout<<"Back element is: "<<age.back();
age.pop();
age.pop()
while(!age.empty())
{
  cout<<"elements in the queue are: "<<
  age.front();
  age.pop;
}
return 0;
}
```

O/P:

Front element is : 21
Back element : 24
Elements in the queue are: 23
Elements in queue are :24