

Problem Statement:

Write a c++ program on function overloading.

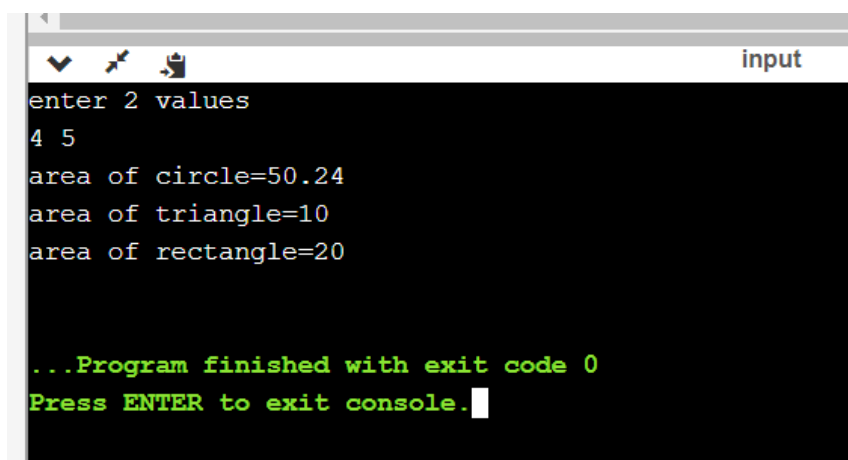
Description: when two or more functions have same name but different parameters is called function overloading .The advantage of function overloading is that it increases the readability of program .

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
void shape(int x);
void shape(int x,int y,float z);
void shape(int x,int y);
void shape(int x)
{
    cout<<"area of circle="<<(3.14*x*x)<<endl;
}
void shape(int x,int y ,float z)
{
    cout<<"area of triangle="<<(z*x*y)<<endl;
}
void shape(int x,int y)
{
    cout<<"area of rectangle="<<(x*y)<<endl;
}
main()
{
    int x,y;
    float z=0.5;
    cout<<"enter 2 values"<<endl;
    cin>>x>>y;
```

```
shape(x);  
shape(x,y,z);  
shape(x,y);  
getch();  
}
```

Output:

A screenshot of a console window with a title bar that says "input". The window has a dark background with white and green text. The output shows the program prompting for two values, receiving "4 5", and then calculating and displaying the areas for a circle, triangle, and rectangle. The program ends with a green message indicating it finished with exit code 0 and a prompt to press ENTER to exit the console.

```
enter 2 values  
4 5  
area of circle=50.24  
area of triangle=10  
area of rectangle=20  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

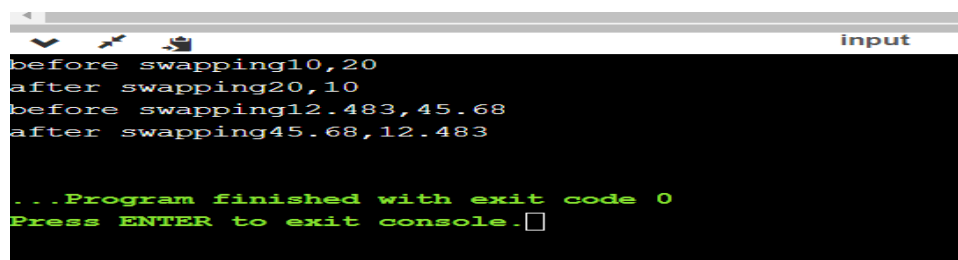
Problem statement:

Write a c++ program to implement function template.

Description: Function templates are special functions that can operate with generic types.

Program:

```
using namespace std;
#include<iostream>
#include<conio.h>
template <class T>
void swap1(T &a,T &b){
    T temp;
    temp=a;
    a=b;
    b=temp;
}int main(){
    int x,y;
    x=10,y=20;
    cout<<"before swapping"<<x<<","<<y<<endl;
    swap1(x,y);
    cout<<"after swapping"<<x<<","<<y<<endl;
    float p=12.483,q=45.68;
    cout<<"before swapping"<<p<<","<<q<<endl;
    swap1(p,q);
    cout<<"after swapping"<<p<<","<<q<<endl;
    return 0;
}
```

Output:

```
input
before swapping10,20
after swapping20,10
before swapping12.483,45.68
after swapping45.68,12.483

...Program finished with exit code 0
Press ENTER to exit console.□
```

problem statement:

Write a c++ program to implement operator overloading.

Description: Operator overloading is the ability to tell the compiler how to perform certain operation when its corresponding operator is used on one or more variables.

Program:

```
#include<iostream>

#include<conio.h>

using namespace std;

class oo
{
    int x,y;

    public: void getdata(int a,int b);

    void display();

    void operator -();

};

void oo::getdata(int a,int b)
{
    x=a;

    y=b;

}

void oo::display()
{
    cout<<"x:"<<x<<endl;

    cout<<"y:"<<y<<endl;

}


void oo::operator -()
{
    x=-x;

    y=-y;

}
```

```
int main()
{
    oo obj;
    obj.getdata(10,-20);
    obj.display();
    -obj;
    obj.display();
}
```

Output:

 C:\Users\Aakanksha\Documents\oo.exe

```
x:10
y:-20
x:-10
y:20

-----
Process exited after 4.623 seconds with return value 0
Press any key to continue . . .
```

Problem statement:

write a c++ program to implement friend function.

Description:


Friend function of a class is defined outside that class scope but it has the right to access all private and protected members of the class.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
class ff
{
    private:int x,y,z;
    public:void getdata(int a,int b);
    void display();
    friend void add(ff &obj);
};
void ff::getdata(int a,int b)
{
    x=a;
    y=b;
}
void ff::display()
{
    cout<<"sum:"<<z<<endl;
}
void add(ff &obj)
{
    obj.z=obj.x+obj.y;
}
int main()
```

```
{  
    ff obj;  
    obj.getdata(10,-20);  
    add(obj);  
    obj.display();  
}
```

Output:

 C:\Users\Aakanksha\Documents\ff.exe

```
sum: -10  
-----  
Process exited after 0.4211 seconds with return value 0  
Press any key to continue . . .
```

Problem statement:

Write a c++ program on class template.

Description: A class template defines a family of class. A class template provides a specification for generating classes based on parameters.

Program: #include<iostream>

#include<conio.h>

using namespace std;

template<class T>

class Max {

 T x, y;

public:

 Max()

 {

 }

 Max(T a, T b) {

 x = a;

 y = b;

 }

 T getMax() {

 if (x > y)

 return x;

 else

 return y;

 }

};

int main() {

 Max <int> i;

 int a, b;

 Max <float> f;

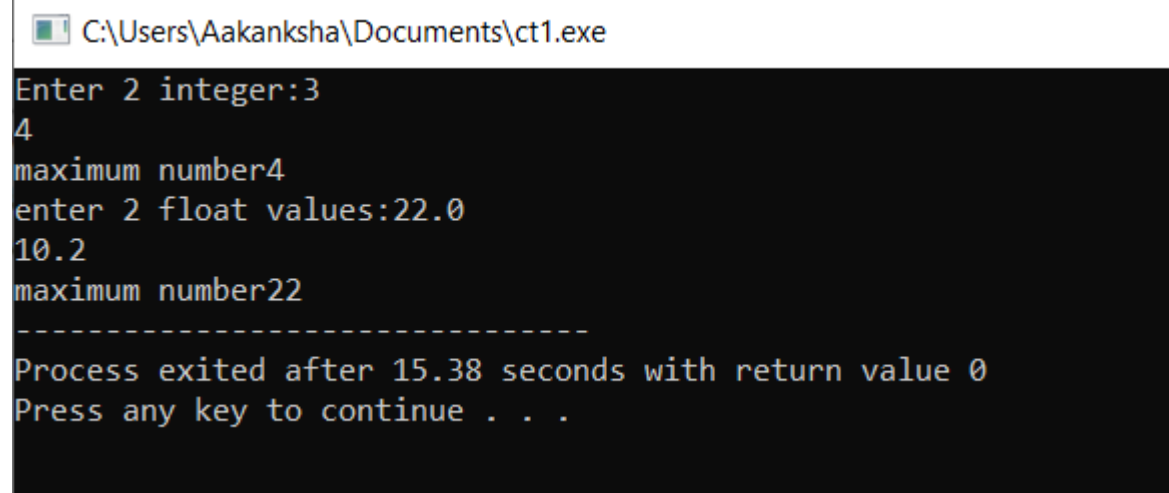
 float c, d;


```

cout << "Enter 2 integer:";
cin >> a>>b;
i = Max<int>(a, b);
cout <<"maximum number"<<i.getMax()<<endl;
cout << "enter 2 float values:";
cin >> c>>d;
f = Max<float>(c, d);
cout <<"maximum number"<<f.getMax();
return 0;
}

```

Output:



```

C:\Users\Aakanksha\Documents\ct1.exe
Enter 2 integer:3
4
maximum number4
enter 2 float values:22.0
10.2
maximum number22
-----
Process exited after 15.38 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a c++ program to implement

- a) single inheritance b) multiple inheritance c) multilevel inheritance d) hybrid inheritance

Description: The process of acquiring the properties of existing class into a new class is called inheritance.

Program:a)single inheritance:Deriving a single subclass from single super class is called single inheritance

```
#include<iostream>

#include<conio.h>

using namespace std;

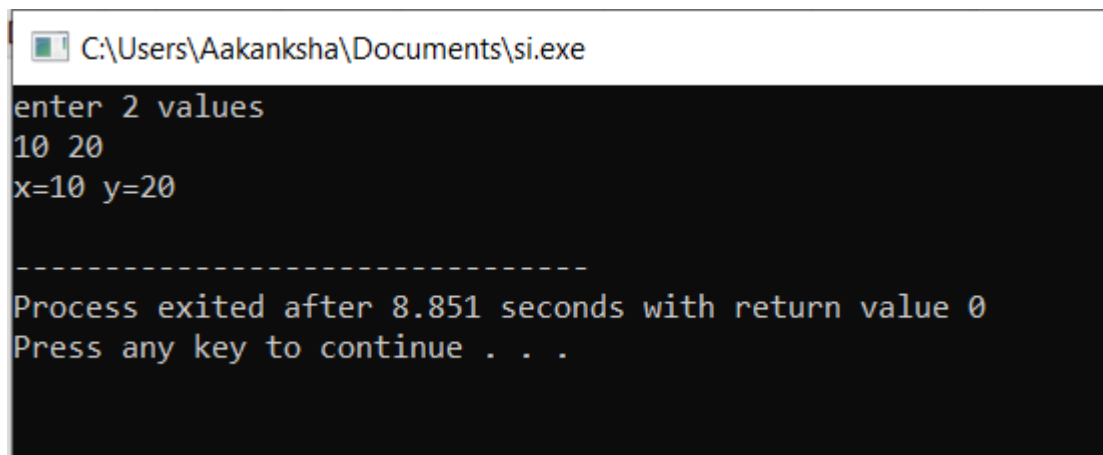
class a
{
    public:int x,y;
    public:void getdata()
    {
        cout<<"enter 2 values"<<endl;
        cin>>x>>y;
    }
};

class b:public a
{
    public:void putdata()
    {
        getdata();
        cout<<"x="<<x<<" "<<"y="<<y<<endl;
    }
};

int main()
{
    b obj;
```

```
        obj.putdata();  
    }
```

Output:



```
C:\Users\Aakanksha\Documents\si.exe  
enter 2 values  
10 20  
x=10 y=20  
  
-----  
Process exited after 8.851 seconds with return value 0  
Press any key to continue . . .
```

b)multiple inheritance:

Description: A derived class with multiple base class is called multiple inheritance.

program:

```
#include<iostream>
#include<conio.h>
using namespace std;
class student1
{
    public:char name1[20];
    int marks1;
    public:void get1()
    {
        cout<<"enter the name and marks"<<endl;
        cin>>name1>>marks1;
    }
};
class student2
{
    public:char name2[20];
    int marks2;
    public:void get2()
    {
        cout<<"enter the name and marks"<<endl;
        cin>>name2>>marks2;
    }
};
class student:public student1,student2
```

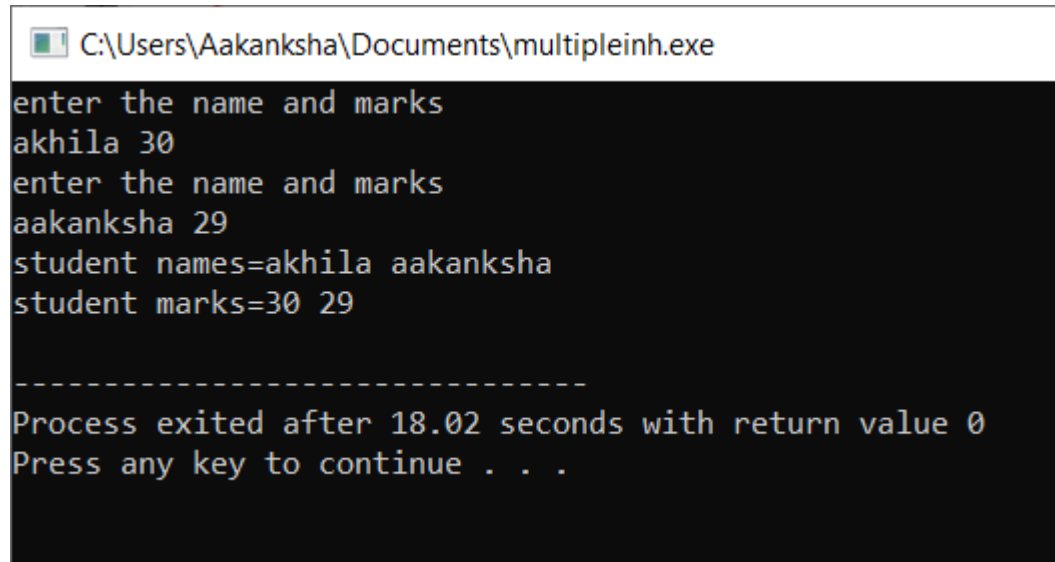
```

{
    public: void put()
    {
        get1();
        get2();
        cout<<"student names="<<name1<<" "<<name2<<endl;
        cout<<"student marks="<<marks1<<" "<<marks2<<endl;
    }
};

int main()
{
    student obj;
    obj.put();
}

```

Output:



```

C:\Users\Aakanksha\Documents\multipleinh.exe
enter the name and marks
akhila 30
enter the name and marks
aakanksha 29
student names=akhila aakanksha
student marks=30 29

-----
Process exited after 18.02 seconds with return value 0
Press any key to continue . . .

```

c)multilevel inheritance

Description: Deriving a single subclass from more than two super class is called multilevel inheritance.

program:

```
#include<iostream>
#include<conio.h>
using namespace std;
class A
{
    public:int x,y,a,b;
    public:void getdata()
    {
        cout<<"enter 2 values"<<endl;
        cin>>x>>y;
    }
    void putdata()
    {
        cout<<"x="<<x<<" "<<"y="<<y<<endl;
    }
};
class B:public A
{
    public:void add()
    {
        a=x+y;
    }
    public:void sub()
```

```

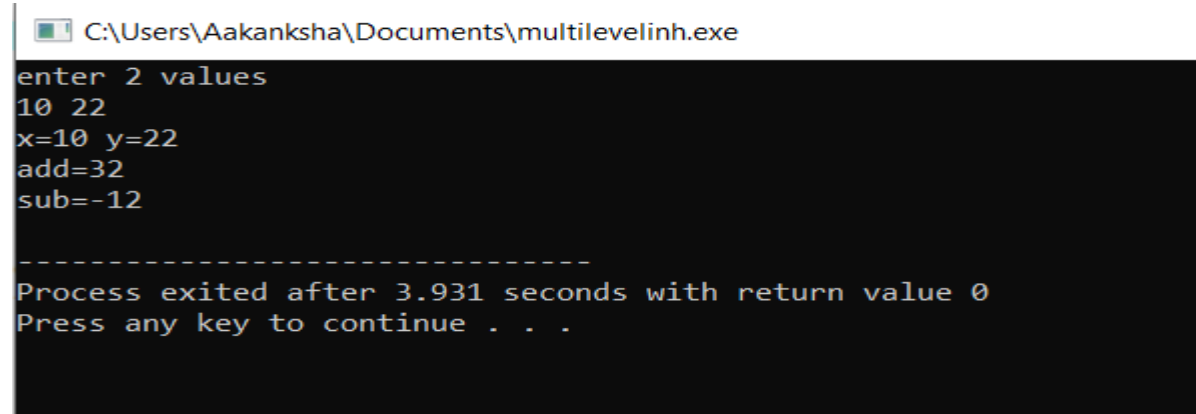
        {
            b=x-y;
        }
};

class C:public B
{
    public:display()
    {
        cout<<"add="<<a<<endl;
        cout<<"sub="<<b<<endl;
    }
};

int main()
{
    C obj;
    obj.getdata();
    obj.putdata();
    obj.add();
    obj.sub();
    obj.display();
}

```

Output:



```

C:\Users\Aakanksha\Documents\multilevelinh.exe
enter 2 values
10 22
x=10 y=22
add=32
sub=-12

-----
Process exited after 3.931 seconds with return value 0
Press any key to continue . . .

```

d)hybrid inheritance:

Description: combination of multiple and hierarchial inheritance is called hybrid inheritance.

program:

```
#include<iostream>
#include<conio.h>
using namespace std;
class A
{
    public:int x,y,a,b;
    public:void getdata()
    {
        cout<<"enter 2 values"<<endl;
        cin>>x>>y;
    }
};
class B:public A
{
    public:void add()
    {
        getdata();
        a=x+y;
        cout<<"add="<<a<<endl;
    }
};
class C:public A
{
    public:void sub()
```

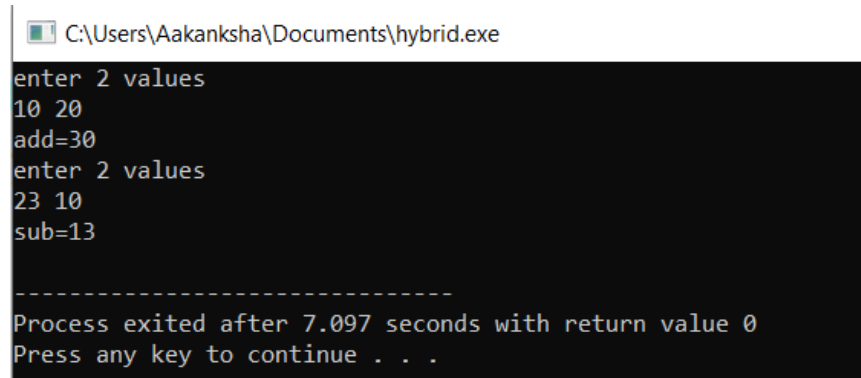


```

        {
            getdata();
            b=x-y;
            cout<<"sub="<<b<<endl;
        }
};
class D:public B,public C
{
    public: void display()
    {
        add();
        sub();
    }
};
int main()
{
    D obj;
    obj.display();
}

```

Output:



```

C:\Users\Aakanksha\Documents\hybrid.exe
enter 2 values
10 20
add=30
enter 2 values
23 10
sub=13

-----
Process exited after 7.097 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a c++ program that uses functions to perform the following operations on a single linked list of integers

a.creation b.insertion c.deletion d.display

Description:

Single linked list: It is a collection of nodes and each node consists of 2 parts called data part and link part.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
class node{
public:
int data;
node *next;
};
node *header=new node;
class LinkedList{
public:
LinkedList(){
header->next=NULL;
}
void creation(){
char ch;
int e;
node *prev;
prev=header;
do{
node *newnode=new node;
```

```

cout<<"enter an element"<<endl;

cin>>e;

newnode->data=e;
newnode->next=NULL;
if(header->next==NULL){
header->next=newnode;
prev=newnode;
}
else{
prev->next=newnode;
prev=newnode;
}

cout<<"do you want to enter another element y /n";
cin>>ch;
}while(ch == 'y' || ch=='Y');
}

void insert_by_pos(){
int el, pos;

cout<<"enter the element u want to insert"<<endl;
cin>>el;

cout<<"enter after at what position u r intrested to insert"<<endl;
cin>>pos;

node *newnode=new node;
if(pos==0){
newnode->data=el;
newnode->next=header->next;
header->next=newnode;
}
else{
node *temp=header->next;
int count=0;

```

```

while(temp->next!=NULL && count<=(pos-1)){
temp=temp->next;
count++;
}
if(temp==NULL){
cout<<"invalid position"<<endl;
}
else{
newnode->data=el;
newnode->next=temp->next;
temp->next=newnode;
}}}
void insert_by_el(){
int el,val;
cout<<"enter element u r entering"<<endl;
cin>>el;
cout<<"enter after what value"<<endl;
cin>>val;
node *newnode=new node;
node *temp=header->next;
while(temp->next!=NULL && temp->data!=val){
temp=temp->next;
}
if(temp==NULL){
cout<<"element not found"<<endl;
}
else{
newnode->data=el;
newnode->next=temp->next;
temp->next=newnode;
}}

```

```

void deletion_of_ele(){
int el;
cout<<"enter what element to be deleted"<<endl;
cin>>el;
node *prev=header;
node *temp=header->next;
while(temp->next!=NULL && temp->data!=el){
prev=prev->next;
temp=temp->next;
}
if(temp==NULL){
cout<<"element not found"<<endl;
}
else{
prev->next=temp->next;
delete(temp);
}}
void display(){
node *temp;
temp=header->next;
while(temp!=NULL ){
cout<<temp->data<<" "<<endl;
temp=temp->next;
}}
void deletion_by_pos(){
int pos;
cout<<"enter at what positioned element to be deleted"<<endl;
cin>>pos;
node *prev=header;
node *temp=header->next;
int count=0;

```

```

while(temp!=NULL && count<=(pos-1)){
prev=prev->next;
temp=temp->next;
count++;
}
if(temp==NULL){
cout<<"element not found"<<endl;
}
else{
prev->next=temp->next;
delete(temp);
}}};

int main(){
int ch;
LinkedList ob;
cout<<"1)create a list"<<endl;
cout<<"2)insert by position"<<endl;
cout<<"3)insert by element"<<endl;
cout<<"4)deletion of element"<<endl;
cout<<"5)display"<<endl;
cout<<"6)delete at position"<<endl;
cout<<"7)exit"<<endl;
do{
cout<<"enter ur choice"<<endl;
cin>>ch;
switch(ch){
case 1:
ob.creation();
break;
case 2:
ob.insert_by_pos();

```

```

break;
case 3:
ob.insert_by_el();
case 4:
ob.deletion_of_ele();
break;
case 5:
ob.display();
break;
case 6:
ob.deletion_by_pos();
break;
case 7:
break;
}}
while(ch!=7);
}

```

Output:

```

3  using namespace std;
4  class node
input
1)create a list
2)insert by position
3)insert by element
4)deletion of element
5)display
6)delete at position
7)exit
enter ur choice
11
enter ur choice
1
enter an element
10
do you want to enter another element y /ny
enter an element
20
do you want to enter another element y /ny
enter an element
40
do you want to enter another element y /nn
enter ur choice
5
10
20
40

```

```
input
enter ur choice
4
enter what element to be deleted
2
enter ur choice
2
enter the element u want to insert
22
enter after at what position u r intrested to insert
2
enter ur choice
5
10
20
22
enter ur choice
6
enter at what positioned element to be deleted
5
element not found
enter ur choice
5
10
20
```

```
input
5
10
22
enter ur choice
6
enter at what positioned element to be deleted
3
element not found
enter ur choice
6
enter at what positioned element to be deleted
5
element not found
enter ur choice
1
enter an element
12
do you want to enter another element y /ny
enter an element
123
do you want to enter another element y /n5
enter ur choice
5
12
123
enter ur choice
```


Problem statement:

Write a c++ template based c++ program that uses functions to perform the following operations on double linked list

a. creation b. insertion c. deletion d. display.

Description:A double linked list is a collection of nodes and each nodes consists of 3 parts called data part ,left link and right link part.

Program:

```
#include<iostream>

#include<conio.h>

using namespace std;

template<class T>

class node{

    public : T data;

    node<T> *prev;

    node<T> *next;

    node()

    {

        this->prev=NULL;

        this->next=NULL;

    }

};

template<class T>

class linkedlist{

    node<T> *head,*tail;

    public:linkedlist(){

        head=NULL;

        tail=NULL;

    }

    void insert_beg(){

        T value;

        cout<<"enter a value"<<endl;

        cin>>value;
```

```

node<T> *newnode=new node<T>;
newnode->data=value;
if(head==NULL || tail==NULL){
    newnode->prev=NULL;
    newnode->next=NULL;
    head=newnode;
    tail=newnode;
}
else{
    newnode->prev=NULL;
    newnode->next=head;
    head->prev=newnode;
    head=newnode;
}
}

void insert_end(){
    T value;
    cout<<"enter a value"<<endl;
    cin>>value;
    node<T> *newnode=new node<T>;
    newnode->data=value;
    if(head==NULL || tail==NULL){
        newnode->prev=NULL;
        newnode->next=NULL;
        head=newnode;
        tail=newnode;
    }
    else{
        newnode->prev=tail;
        newnode->next=NULL;
        tail->next=newnode;
        tail=newnode;
    }
}

```

```

}}

void insert_pos(){
    T value,pos;
    cout<<"enter a value"<<endl;
    cin>>value;
    cout<<"enter the position to place the new element"<<endl;
    cin>>pos;
    node<T> *newnode=new node<T>;
    newnode->data=value;
    node<T> *temp=head;
    for(int i=1;i<pos-1;i++){
        temp=temp->next;
    }
    if(temp->next==NULL){
        newnode->next=NULL;
        newnode->prev=temp;
        head=newnode;
    }
    else{
        node<T> *temp1=temp->next;
        newnode->next=temp1;
        temp1->prev=newnode;
        newnode->prev=temp;
        temp->next=newnode;
    }
}

void delete_beg(){
    if(head==NULL){
        cout<<"deletion is not possible"<<endl;
        exit(1);
    }
    if((head!=NULL)&&(head==tail)){

```

```

        node<T> *temp=head;
        head=NULL;
        tail=NULL;
        cout<<"deleted node info"<<temp->data<<endl;
        delete(temp);
    }
    else{
        node<T> *temp=head;
        temp->next->prev=NULL;
        cout<<"deleted node info"<<temp->data<<endl;
        head=temp->next;
        delete(temp);
    }
}

void delete_end(){
    if(head==NULL){
        cout<<"deletion is not possible"<<endl;
        exit(1);
    }
    if((head!=NULL)&&(head==tail)){
        node<T> *temp=head;
        head=NULL;
        tail=NULL;
        cout<<"deleted node info"<<temp->data<<endl;
        delete(temp);
    }
    else{
        node<T> *temp=tail;
        temp->prev->next=NULL;
        tail=temp->prev;
        cout<<"deleted node info"<<temp->data<<endl;
        delete(temp);
    }
}

```

```

    }}

void delete_pos(){
    int pos,i;

    cout<<"enter the position to delete the element"<<endl;
    cin>>pos;
    node<T> *temp=head;
    node<T> *temp1,*temp2;
    while(temp->next!=NULL){
        for(i=0;i<pos-1;i++)
            temp=temp->next;
        if(i==pos-1)
            break;
    }
    cout<<"deleted node info"<<temp->data<<endl;
    if(temp==head){
        temp1=temp->next;
        temp1->prev=NULL;
        head=temp1;
    }
    else{
        temp1=temp->prev;
        temp2=temp->next;
        temp1->next=temp2;
        temp2->prev=temp1;
    }
}

void ftraverse(){
    node<T> *temp=head;
    while(temp!=NULL){
        cout<<"forward traverse:"<<temp->data<<endl;
        temp=temp->next;
    }
}

```

```

void rtraverse(){
    node<T> *temp=tail;
    while(temp!=NULL){
        cout<<"reverse traverse:"<<temp->data<<endl;
        temp=temp->prev;
    }
};

```

```

int main(){
    linkedlist<int> obj;
    int choice;

    cout<<"enter 1 to insert at beg"<<endl;
    cout<<"enter 2 to insert at end"<<endl;
    cout<<"enter 3 to delete at beg"<<endl;
    cout<<"enter 4 to delete at end"<<endl;
    cout<<"enter 5 to forward traverse"<<endl;
    cout<<"enter 6 to reverse traverse"<<endl;
    cout<<"enter 7 to insert at position"<<endl;
    cout<<"enter 8 to delete at position"<<endl;

    while(1){
        cout<<"enter choice"<<endl;
        cin>>choice;
        switch(choice){
            case 1:obj.insert_beg();
            break;
            case 2:obj.insert_end();
            break;
            case 3:obj.delete_beg();
            break;
            case 4:obj.delete_end();
            break;
            case 5:obj.ftraverse();
            break;

```

```

        case 6:obj.rtraverse();

                break;

        case 7:obj.insert_pos();

                break;

        case 8:obj.delete_pos();

                break;

        case 9:exit(1);

        break

    }}}

```

Output:

```

C:\Users\Aakanksha\Documents\tdll.exe
enter 1 to insert at beg
enter 2 to insert at end
enter 3 to delete at beg
enter 4 to delete at end
enter 5 to forward traverse
enter 6 to reverse traverse
enter 7 to insert at position
enter 8 to delete at position
enter choice
1
enter a value
10
enter choice
2
enter a value
20
enter choice
7
enter a value
30
enter the position to place the new element
2
enter choice
5
forward traverse:10
forward traverse:30
forward traverse:20
enter choice

enter choice
3
deleted node info10
enter choice
4
deleted node info20
enter choice
5
forward traverse:30
enter choice
1
enter a value
60
enter choice
6
reverse traverse:30
reverse traverse:60
enter choice

```

Problem statement:

write a c++ program to implement stack ADT using an array and a linked list

Description: stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO and FILO

program: stack using array

```
#include<iostream>

#include<conio.h>

#define size 100

using namespace std;

class stack {

    public:

    int a[size];

    int top;

    public: void create();

    void push(int value);

    void pop();

    void display();

    void peak();

};

void stack::create(){

    top=-1;

}

void stack::push(int value){

    if(top==(size-1)){

        cout<<"stack is full!!push is not possible"<<endl;

    }

    else{

        top++;

        a[top]=value;

    }

}
```



```

    }}
void stack::pop(){
    if(top==-1){
        cout<<"stack is empty !pop is not possible"<<endl;
    }
    else{
        cout<<"deleted element"<<a[top]<<endl;
        top--;
    }
}
void stack::display(){
    int i=0;
    for(i=0;i<=top;i++){
        cout<<a[i]<<" "<<endl;
    }
}
void stack::peak(){
    if(top>-1){
        cout<<"top element"<<a[top]<<endl;
    }
}
int main(){
    int ch;
    stack obj;
    obj.create();
    cout<<"1)push"<<endl;
    cout<<"2)pop"<<endl;
    cout<<"3)display"<<endl;
    cout<<"4)peak"<<endl;
    cout<<"7)exit"<<endl;
    do{
        cout<<"enter ur choice"<<endl;

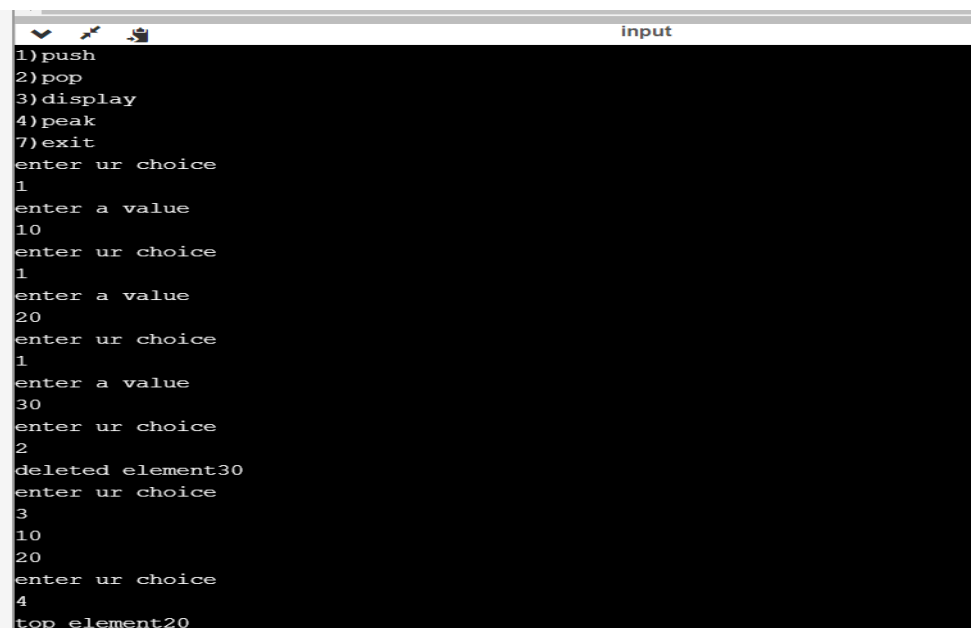
```

```

cin>>ch;
switch(ch){
case 1:
    int value;
    cout<<"enter a value"<<endl;
    cin>>value;
case 1:obj.push(value);
break;
case 2:obj.pop();
break;
case 3:obj.display();
break;
case 4:obj.peak();
break;
default:cout<<"plz enter valid option"<<endl;
break;
}while(ch!=4);
}

```

Output:



```

input
1)push
2)pop
3)display
4)peak
7)exit
enter ur choice
1
enter a value
10
enter ur choice
1
enter a value
20
enter ur choice
1
enter a value
30
enter ur choice
2
deleted element30
enter ur choice
3
10
20
enter ur choice
4
top element20

```

Program: stack with linked list

```
#include<iostream>
#include<conio.h>
#define size 100
using namespace std;
class node{
    public:
    int data;
    node *next;
};
node *front=new node;
class stack{
    public:
    stack(){
        front=NULL;
    }
    void push();
    void pop();
    void peak();
    void display();
};
void stack::push(){
    int value;
    cout<<"enter element u r entering"<<endl;
    cin>>value;
    node *temp=new node();
    temp->data=value;
    if(front==NULL){
        temp->next=NULL;
```

```

    }else{
        temp->next=front;
    }
    front=temp;
}

void stack::pop(){
    if(front==NULL){
        cout<<"stack is empty !pop is not possible"<<endl;
    }else{
        node *temp=front;
        front=front->next;
        cout<<"deleted element"<<front->data<<endl;
        delete(temp);
    }
}

void stack::display(){
    node *temp=front;
    while(temp!=NULL){
        cout<<temp->data<<" "<<endl;
        temp=temp->next;
    }
}

void stack::peak(){
    cout<<"top element"<<front->data<<endl;
}

int main(){
    int ch;
    stack obj;
    cout<<"1)push"<<endl;
    cout<<"2)pop"<<endl;
    cout<<"3)display"<<endl;

```

```

cout<<"4)peak"<<endl;
cout<<"7)exit"<<endl;
do{
cout<<"enter ur choice"<<endl;
cin>>ch;
switch(ch){
case 1:obj.push();
break;
case 2:obj.pop();
break;
case 3:obj.display();
break;
case 4:obj.peak();
break;
default:cout<<"plz enter valid option"<<endl;
break;
}while(ch!=4);
}

```

Output:

```

input
1)push
2)pop
3)display
4)peak
7)exit
enter ur choice
1
enter element u r entering
10
enter ur choice
1
enter element u r entering
22
enter ur choice
1
enter element u r entering
12
enter ur choice
3
12
22
10
enter ur choice
2
deleted element22
enter ur choice

```

Problem statement:

Write a c++ program that uses stack operations to convert a given infix expression to postfix equivalent. Implement the stack using array.

Description: To convert infix to postfix expression, we will use the stack data structure. By scanning the infix expression from left to right, when we will get any operand simply add them to postfix and operator and parenthesis add them to stack maintain the precedence of them.

Program:

```
#include<iostream>
#include<conio.h>
#include<string.h>
#include<stack>
using namespace std;
int prec(char c) {
    if(c == '^')
        return 3;
    else if(c == '*' || c == '/')
        return 2;
    else if(c == '+' || c == '-')
        return 1;
    else
        return -1;
}
void infixToPostfix(string s){
    stack<char> st;
    st.push('N');
    int l = s.length();
    string ns;
    for(int i = 0; i < l; i++) {
        if((s[i] >= 'a' && s[i] <= 'z') || (s[i] >= 'A' && s[i] <= 'Z'))
            ns+=s[i];
    }
```

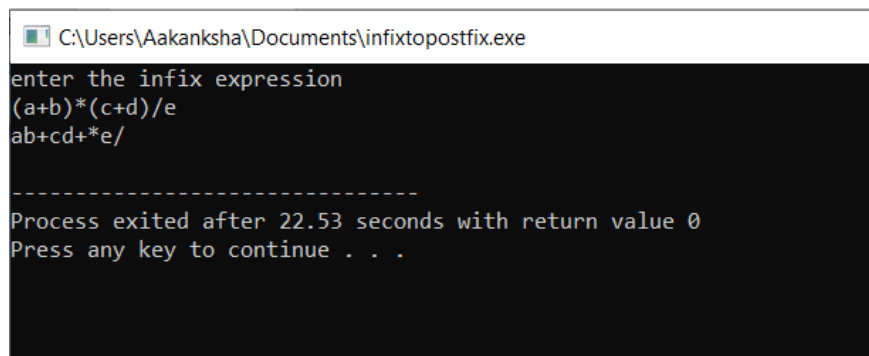
```

else if(s[i] == '{'){
    st.push('{');
}
else if(s[i] == '){
    while(st.top() != 'N' && st.top() != '{'){
        char c = st.top();
        st.pop();
        ns += c;
    }
    if(st.top() == '{'){
        char c = st.top();
        st.pop();
    }
}
else{
    while(st.top() != 'N' && prec(s[i]) <= prec(st.top())) {
        char c = st.top();
        st.pop();
        ns += c;
    }
    st.push(s[i]);
}
}
while(st.top() != 'N') {
    char c = st.top();
    st.pop();
    ns += c;
}
cout << ns << endl;
}
int main(){

```

```
string exp;  
cout<<"enter the infix expression"<<endl;  
cin>>exp;  
infixToPostfix(exp);  
return 0;  
}
```

Output:



```
C:\Users\Aakanksha\Documents\infixtopostfix.exe  
enter the infix expression  
(a+b)*(c+d)/e  
ab+cd+*e/  
-----  
Process exited after 22.53 seconds with return value 0  
Press any key to continue . . .
```


Problem statement:

Write a c++ program to implement a queue ADT using an array and linked list.

Description: A queue is a linear data structure which follows a particular order in which the operations are performed. The order is FIFO.

Program: queue using array

```
#include<iostream>

#include<conio.h>

#define size 100

using namespace std;

class queue {
    public:
        int a[size];
        int front,rear;
        public: void create();
        void enqueue(int value);
        void dequeue();
        void display();
        void peak();
};

void queue::create(){
    front=rear=-1;
}

void queue::enqueue(int value){
    if(rear==(size-1)){
        cout<<"queue is full!!enqueue is not possible"<<endl;
    }
    else if(rear==size-1 || front==size-1){
        front=rear=0;
        a[rear]=value;
    }
}
```

```

else{
    rear++;
    a[rear]=value;
}
}
void queue::dequeue(){
    if(front==-1){
        cout<<"queue is empty !dequeue is not possible"<<endl;
    }
    else{
        cout<<"deleted element"<<a[front]<<endl;
        front++;
        if(front==rear){
            front=rear=-1;
        }
    }
}
void queue::display(){
    int i=0;
    for(i=front;i<=rear;i++){
        cout<<"elements:"<<a[i]<<" "<<endl;
    }
}
int main(){
    int ch;
    queue obj;
    obj.create();
    cout<<"1)enqueue"<<endl;
    cout<<"2)dequeue"<<endl;
    cout<<"3)display"<<endl;
    cout<<"4)exit"<<endl;
    do{
        cout<<"enter ur choice"<<endl;

```

```

cin>>ch;
switch(ch){
case 1: int value;
        cout<<"enter a value"<<endl;
        cin>>value;
        obj.enqueue(value);
        break;
case 2:obj.dequeue();
        break;
case 3:obj.display();
        break;
case 4:exit(1);
        break;
default:cout<<"plz enter valid option"<<endl;
        break;
}while(ch!=4);
}

```

Output:

```

C:\Users\Aakanksha\Documents\aqueue.exe
1)enqueue
2)dequeue
3)display
4)exit
enter ur choice
1
enter a value
10
enter ur choice
1
enter a value
20
enter ur choice
1
enter a value
30
enter ur choice
3
elements:10
elements:20
elements:30
enter ur choice
2
deleted element10
enter ur choice
3
elements:20
elements:30
enter ur choice

```

Program: queue using linked list

```
#include<iostream>
#include<conio.h>
#define size 100
using namespace std;
class node{
    public:
    int data;
    node *next;
};
node *front=new node;
node *rear=new node;
class queue{
    public:
    queue(){
        front=rear=NULL;
    }
    void enqueue();
    void dequeue();
    void display();
};
void queue::enqueue(){
    int value;
    cout<<"enter element u r entering"<<endl;
    cin>>value;
    node *newnode=new node();
    newnode->data=value;
    newnode->next=NULL;
    if(front==NULL){
```

```

        front=rear=newnode;
    }
    else{
        rear->next=newnode;
        rear=newnode;
    }
}

void queue::dequeue(){
    if(front==NULL){
        cout<<"queue is empty !dequeue is not possible"<<endl;
    }
    else{
        node *temp=front;
        cout<<"deleted element"<<front->data<<endl;
        front=front->next;
        delete(temp);
    }
}

void queue::display(){
    node *temp=front;
    while(temp!=NULL){
        cout<<temp->data<<" "<<endl;
        temp=temp->next;
    }
}

int main(){
    int ch;
    queue obj;
    cout<<"1)enqueue"<<endl;
    cout<<"2)dequeue"<<endl;
    cout<<"3)display"<<endl;
    cout<<"7)exit"<<endl;

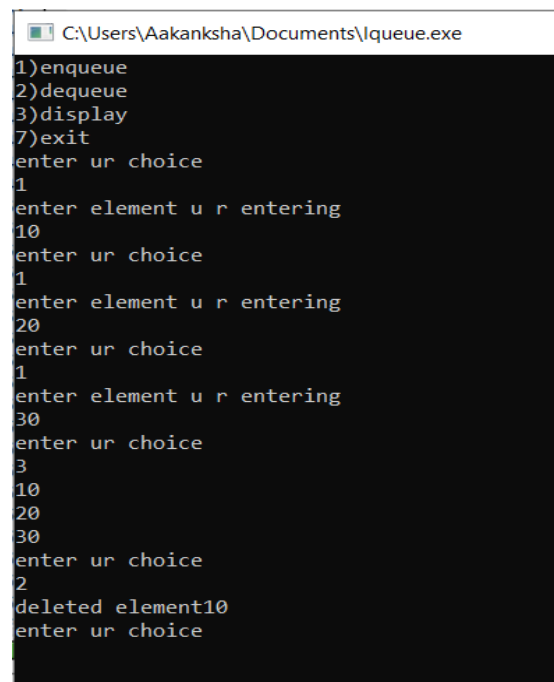
```

```

do{
cout<<"enter ur choice"<<endl;
cin>>ch;
switch(ch){
case 1:obj.enqueue();
break;
case 2:obj.dequeue();
break;
case 3:obj.display();
break;
case 4:exit(1);
break;
default:cout<<"plz enter valid option"<<endl;
break;
}while(ch!=4);
}

```

Output:



```

C:\Users\Aakanksha\Documents\lqueue.exe
1)enqueue
2)dequeue
3)display
7)exit
enter ur choice
1
enter element u r entering
10
enter ur choice
1
enter element u r entering
20
enter ur choice
1
enter element u r entering
30
enter ur choice
3
10
20
30
enter ur choice
2
deleted element10
enter ur choice

```

Problem statement:

Write a c++ program that uses function template to perform search a key element in a list using linear search.

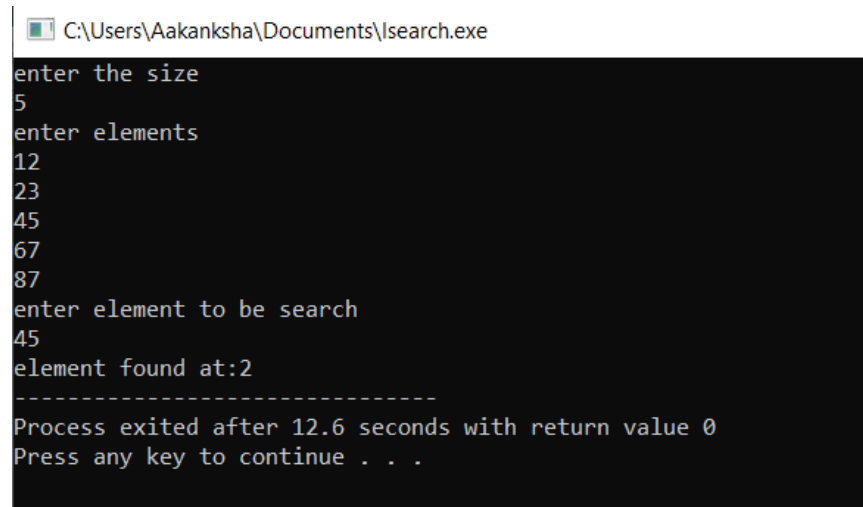
Description: linear search is one of the simple searching technique. It searches the elements one after the other till the element is found. The time complexity is $O(n)$.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
int flag=0;
template<class T>
int lsearch(T a[],int n,T ele){
    int i;
    for(i=0;i<n;i++){
        if(a[i]==ele){
            flag=1;
            return (i);
        }
    }
}
int main(){
    int pos,n,x;
    cout<<"enter the size"<<endl;
    cin>>n;
    int a[n];
    cout<<"enter elements"<<endl;
    for(int j=0;j<n;j++){
        cin>>a[j];
    }
    cout<<"enter element to be search"<<endl;
    cin>>x;
    pos=lsearch(a,n,x);
```

```
    if(flag==1)
        cout<<"element found at:"<<pos;
    else
        cout<<"element not found";
}
```

Output:



```
C:\Users\Aakanksha\Documents\lsearch.exe
enter the size
5
enter elements
12
23
45
67
87
enter element to be search
45
element found at:2
-----
Process exited after 12.6 seconds with return value 0
Press any key to continue . . .
```


Problem statement:

Write a c++ program that uses function template to perform search a key element in a list using binary search.

Description: Binary search uses divide and conquer algorithm. It is one of the fastest searching technique. It divides the list into sub parts till the element is found.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
template<class T>
int bsearch(T a[],int l,int r,T ele)
{
    int mid=(l+r)/2;
    if(l<r)
    {
        if(ele<a[mid])
        {
            bsearch(a,l,mid-1,ele);
        }
        else if(ele>a[mid])
        {
            bsearch(a,mid+1,r,ele);
        }
        else
        {
            return mid;
        }
    }
    else
        return -1;
```

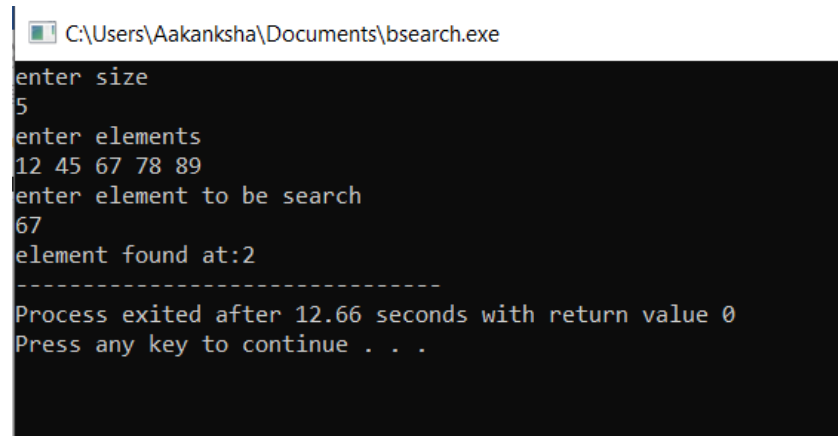
```

}

int main()
{
    int pos,x,n;
    cout<<"enter size"<<endl;
    cin>>n;
    int a[n];
        cout<<"enter elements"<<endl;
    for(int j=0;j<n;j++)
    {
        cin>>a[j];
    }
    cout<<"enter element to be search"<<endl;
    cin>>x;
    pos=(bsearch(a,0,n,x));
    if(pos!=-1)
        cout<<"element found at:"<<pos;
    else
        cout<<"element not found";
}

```

Output:



```

C:\Users\Aakanksha\Documents\bsearch.exe
enter size
5
enter elements
12 45 67 78 89
enter element to be search
67
element found at:2
-----
Process exited after 12.66 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a c++ program that uses function template to arrange a list of elements in ascending order using insertion sort.

Description: Insertion sort is one of the sorting technique. It is similar to the way of sorting playing cards in our hands. It iterates from 0 to n over the list.

Program:

```
#include<iostream>

#include<conio.h>

using namespace std;

template<class T>
int insertionsort(T a[],int n)
{
    int i,j;
    T temp;
    for(i=1;i<n;i++)
    {
        temp=a[i];
        j=i-1;
        while((j>=0)&&(a[j]>temp))
        {
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=temp;
    }
}

int main()
{
    int a[100],n;
```

```

        cout<<"enter the number of elements"<<endl;

        cin>>n;

        cout<<"enter elements"<<endl;
        for(int i=0;i<n;i++)
        {

                cin>>a[i];

        }

        cout<<"before sorting:"<<endl;
        for(int i=0;i<n;i++)
        {

                cout<<a[i]<<" ";

        }

        cout<<endl;

        insertionsort(a,n);

        cout<<"After sorting:"<<endl;

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

                cout<<a[i]<<" ";

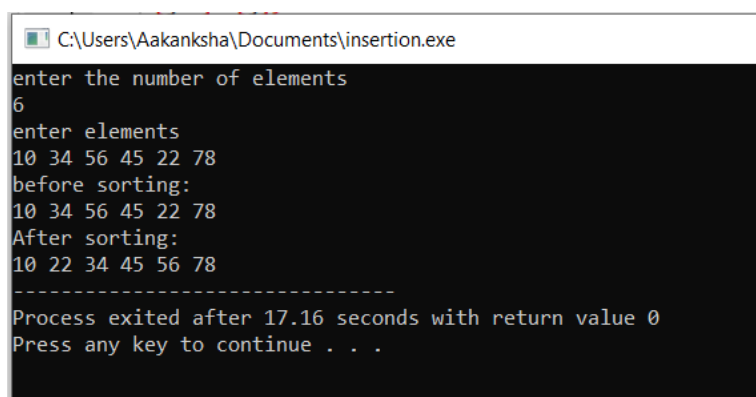
        }

        return 0;

}

```

Output:



```

C:\Users\Aakanksha\Documents\insertion.exe
enter the number of elements
6
enter elements
10 34 56 45 22 78
before sorting:
10 34 56 45 22 78
After sorting:
10 22 34 45 56 78
-----
Process exited after 17.16 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a c++ program that implements selection sort algorithm to arrange a list of elements in descending order.

Description: Selection sort is a simple sorting technique used to sort the elements in the given list. The list is divided into two parts i.e. the sorted part at left end and unsorted part at right end.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
template<class T>
int selectionsort(T a[],int n)
{
    int min,i,j;
    T temp;
    for(i=0;i<n;i++)
    {
        min=i;
        for(j=i+1;j<n;j++)
        {
            if(a[j]>a[min])
                min=j;
        }
        temp=a[i];
        a[i]=a[min];
        a[min]=temp;
    }
}
int main()
{
    int a[100],n;
```

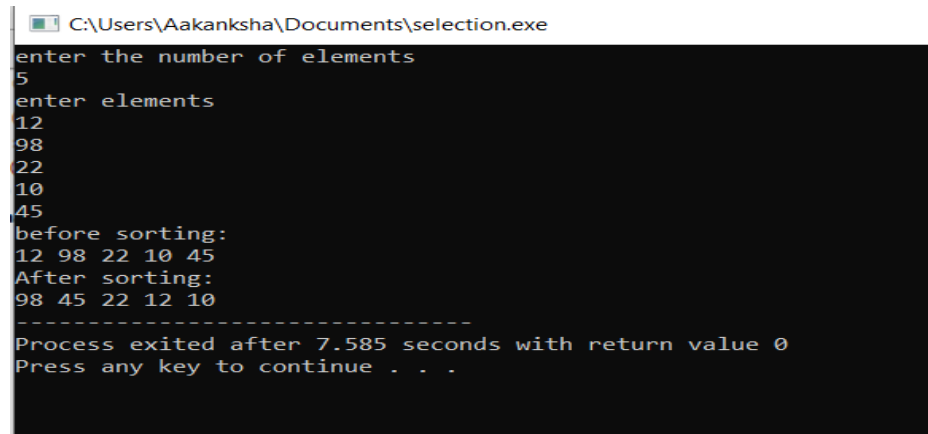
```

        cout<<"enter the number of elements"<<endl;
cin>>n;
cout<<"enter elements"<<endl;
for(int i=0;i<n;i++)
{
    cin>>a[i];
}

cout<<"before sorting:"<<endl;
for(int i=0;i<n;i++)
{
    cout<<a[i]<<" ";
}
cout<<endl;
selectionsort(a,n);
cout<<"After sorting:"<<endl;
for(int i=0;i<n;i++)
{
    cout<<a[i]<<" ";
}
}

```

Output:



```

C:\Users\Aakanksha\Documents\selection.exe
enter the number of elements
5
enter elements
12
98
22
10
45
before sorting:
12 98 22 10 45
After sorting:
98 45 22 12 10
-----
Process exited after 7.585 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a c++program that implements heap sort algorithm for sorting a list of integers in ascending order.

Description: Heap sort is a comparison based sorting technique based on binary heap data structure. Heap sort algorithm uses one of the tree concepts called heap tree. Max heap for descending order and min heap for ascending order.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
void heapcreate(int a[],int n)
{
    int i,j,k,item;
    for(k=1;k<n;k++)
    {
        item=a[k];
        i=k,j=(i-1)/2;
        while(i>0&&item>a[j])
        {
            a[i]=a[j];
            i=j;
            j=(i-1)/2;
        }
        a[i]=item;
    }
}
void adjust(int a[],int n)
{
    int i,j,item;
    j=0;
```

```

    item=a[j];
    i=2*j+1;
    while(i<=n-1)
    {
        if(i+1<=n-1)
            if(a[i]<a[i+1])
                i++;
        if(item<a[i])
        {
            a[j]=a[i];
            j=i;
            i=2*j+1;
        }
        else
            break;
    }
    a[j]=item;
}

void heapsort(int a[],int n)
{
    int i,temp;
    heapcreate(a,n);
    for(i=n-1;i>0;i--)
    {
        temp=a[0];
        a[0]=a[i];
        a[i]=temp;
        adjust(a,i);
    }
}

```



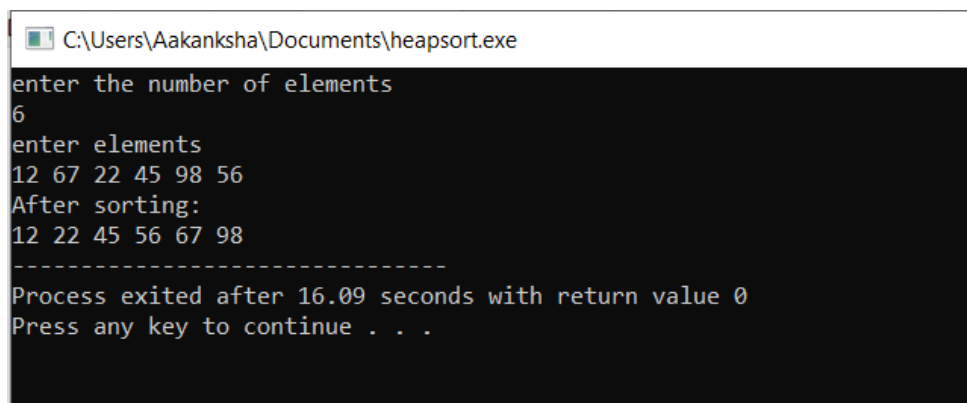
```

}
int main()
{
    int a[100],n;
    cout<<"enter the number of elements"<<endl;
    cin>>n;
    cout<<"enter elements"<<endl;
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
    }
    heapsort(a,n);
    cout<<"After sorting:"<<endl;
    for(int i=0;i<n;i++)
    {
        cout<<a[i]<<" ";
    }

}

```

Output:



```

C:\Users\Aakanksha\Documents\heapsort.exe
enter the number of elements
6
enter elements
12 67 22 45 98 56
After sorting:
12 22 45 56 67 98
-----
Process exited after 16.09 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a template based c++ program that implements quick sort algorithm to arrange a list of elements in ascending order.

Description: Quick sort is the fastest sorting algorithm. It uses divide and conquer algorithm with a special kind of recursion. An element is selected from the list and rearranges the remaining elements around it. Selected element is called "pivot".

Program:

```
#include<iostream>

#include<conio.h>

using namespace std;

template<class T>

void Qsort(T a[],int low,int high){

    T pivot,i,j,temp;

    if(low<high){

        pivot=a[low];

        i=low+1;

        j=high;

        while(1){

            while((pivot>a[i])&&(i<=high))

                {

                    i++;

                }

            while((pivot<a[j])&&(j>=low)){

                j--;

            }

            if(i<j){

                temp=a[i];

                a[i]=a[j];

                a[j]=temp;

            }

        }

    }

}
```

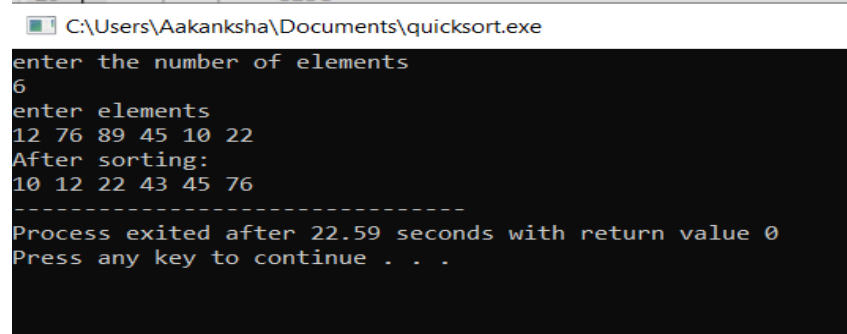
```

        else{
            break;
        }
        a[low]=a[j];
        a[j]=pivot;
        Qsort(a,low,j-1);
        Qsort(a,j+1,high);
    }
}

int main(){
    int n;
    cout<<"enter the number of elements"<<endl;
    cin>>n;
    int arr[n];
    cout<<"enter elements"<<endl;
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    Qsort(arr,0,n);
    cout<<"After sorting:"<<endl;
    for(int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
}

```

Output:



```

C:\Users\Aakanksha\Documents\quicksort.exe
enter the number of elements
6
enter elements
12 76 89 45 10 22
After sorting:
10 12 22 43 45 76
-----
Process exited after 22.59 seconds with return value 0
Press any key to continue . . .

```

Problem statement:

Write a c++ program that implements merge sort algorithm for sorting a list of integers in ascending order.

Description: Merge sort is based on divide and conquer strategy. It is an external sorting technique. It divide the list into two parts of equal size. Repeat the process till there is only one element in the list.

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
void merge(int a[],int l,int mid,int h){
    int i,j,k;
    int b[20];
    i=l;
    j=mid+1;
    k=l;
    while(i<=mid&& j<=h){
        if(a[i]<=a[j]){
            b[k]=a[i];
            i++;
        }
        else{
            b[k]=a[j];
            j++;
        }
        k++;
    }
    if(i>mid){
        for(int n=j;n<=h;n++)
            b[k++]=a[n];
    }
}
```

```

        }else{
            for(int n=i;n<=mid;n++){
                b[k++]=a[n];
            }
            for(k=l;k<=h;k++){
                a[k]=b[k];
            }
        }
    }

void Msort(int a[],int l,int h){
    int mid;
    if(l<h){
        mid=(l+h)/2;
        Msort(a,l,mid);
        Msort(a,mid+1,h);
        merge(a,l,mid,h);
    }
}

int main(){
    int a[100],n;

    cout<<"enter the number of elements"<<endl;

    cin>>n;

    cout<<"enter elements"<<endl;

    for(int i=0;i<n;i++){
        cin>>a[i];
    }

    Msort(a,0,n);

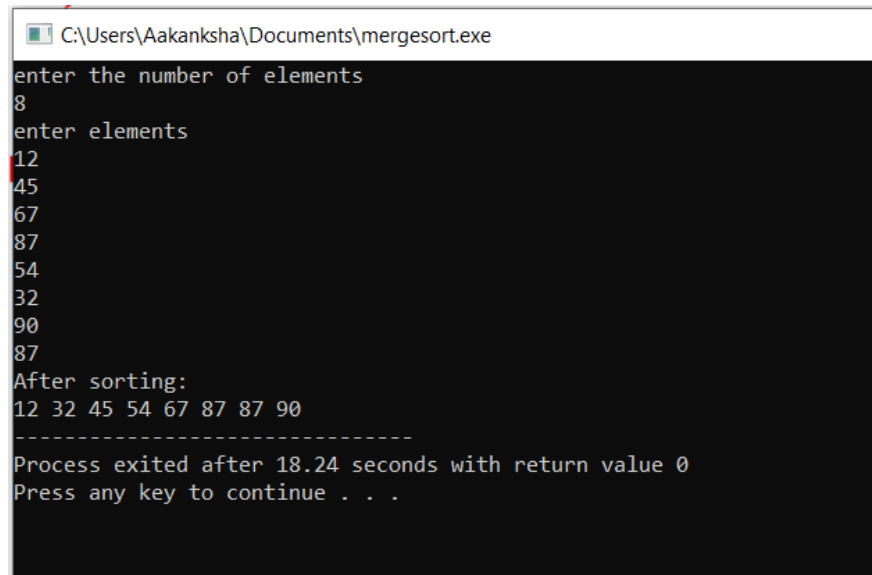
    cout<<"After sorting:"<<endl;

    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }

}

```

Output:



```
C:\Users\Aakanksha\Documents\mergesort.exe
enter the number of elements
8
enter elements
12
45
67
87
54
32
90
87
After sorting:
12 32 45 54 67 87 87 90
-----
Process exited after 18.24 seconds with return value 0
Press any key to continue . . .
```

Problem statement:

Write a c++ program to implement all the functions of a dictionary(ADT) using hashing

Description: Dictionary is the collection of elements with distinct keys. A dictionary is also called hash, a map, a hash map in different programming language. It contains key-value pairs in it.

Program:

```
#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Pair{

    public:int k,v;

};

class Dictionary{

    Pair **ht;

    int size;

    public:Dictionary( ){

        size=7;

        ht=new Pair*[size];

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

            ht[i]=NULL;

    }

    Dictionary(int s){

        size=s;

        ht=new Pair*[size];

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

            ht[i]=NULL;

    }

    public:void insert(int key,int value);

    int search(int key);

    void display();

};
```

```

};

void Dictionary::insert(int key,int value){
    int count=0;
    int hb=key%size;
    int j=hb;
    do{
        if(ht[j]!=NULL){
            j=(j+1)%size;
            count++;
        }
        else{
            break;
        }
    }while(hb!=j);
    if(hb==j&&count!=0){
        cout<<"hash table is full"<<endl;
    }
    else{
        ht[j]=new Pair;
        ht[j]->k=key;
        ht[j]->v=value;
    }
}

int Dictionary::search(int key){
    int hb=key%size;
    int j=hb;
    do{
        if(ht[j]==NULL)
            return -1;
        else if(ht[j]->k==key)

```



```

        return ht[j]->v;
    else
        j=(j+1)%size;
    }while(j!=hb);
    return -1;
}

void Dictionary::display(){
    for(int i=0;i<size;i++){
        cout<<"["<<i<<"]";
        if(ht[i]==NULL)
            cout<<"\n";
        else
            cout<<"->{"<<ht[i]->k<<","<<ht[i]->v<<"}"<<endl;
    }
}

int main(){
    Dictionary ob(11);
    int key,value,l,k;
    int choice;
    while(1){
        cout<<"1.insert"<<endl;
        cout<<"2.search"<<endl;
        cout<<"3.display"<<endl;
        cin>>choice;
        switch(choice){
            case 1:cout<<"enter key and value"<<endl;
                cin>>key>>value;
                ob.insert(key,value);
                break;
            case 2:cout<<"enter the key to search"<<endl;

```

```

        cin>>k;
        l=ob.search(k);
        if(l!=-1){
            cout<<"not found"<<endl;
        }

        else{
            cout<<"value is:"<<l<<endl;
        }

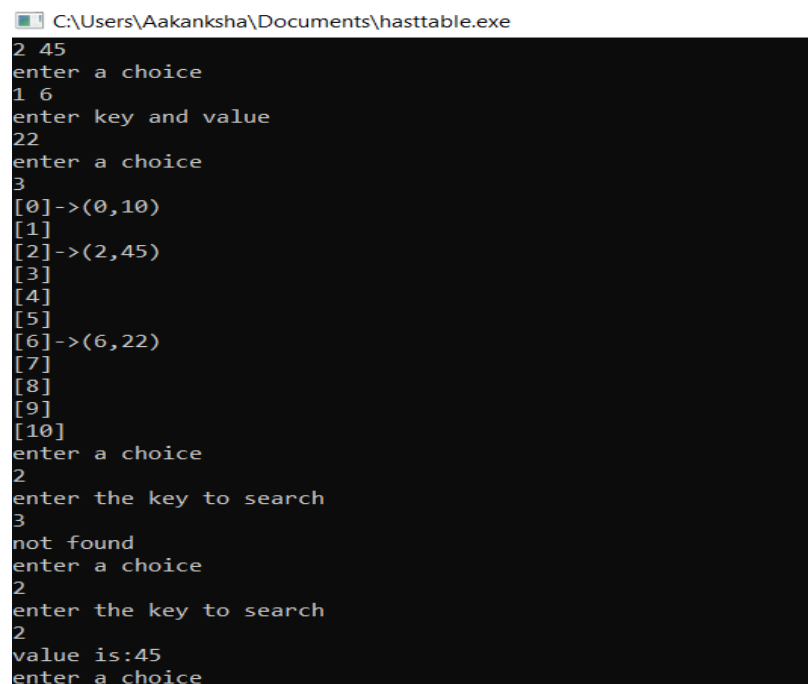
        break;
        case 3:ob.display();
        break;
        case 4:exit(1);
        break;

    }}

    return 0;
}

```

Output:



```

C:\Users\Aakanksha\Documents\hashtable.exe
2 45
enter a choice
1 6
enter key and value
22
enter a choice
3
[0]->(0,10)
[1]
[2]->(2,45)
[3]
[4]
[5]
[6]->(6,22)
[7]
[8]
[9]
[10]
enter a choice
2
enter the key to search
3
not found
enter a choice
2
enter the key to search
2
value is:45
enter a choice

```

Problem statement:

Write a c++ program to traverse a given graph using BFS algorithm.

Description: Breadth first traversal is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighboring nodes.

Program:

```
#include<iostream>

#include<list>

using namespace std;

class graph{
    int v;
    list<int> *adj;
    public:graph(int v){
        this->v=v;
        adj=new list<int>[v];
    }
    void addedge(int v,int e);
    void bfs(int s);
};

void graph::addedge(int v,int e){
    adj[v].push_back(e);
}

void graph::bfs(int s){
    bool *visited=new bool[v];
    for(int i=0;i<v;i++){
        visited[i]=false;
    }
    list<int> queue;
    visited[s]=true;
    queue.push_back(s);
    list<int>::iterator i;
```

```

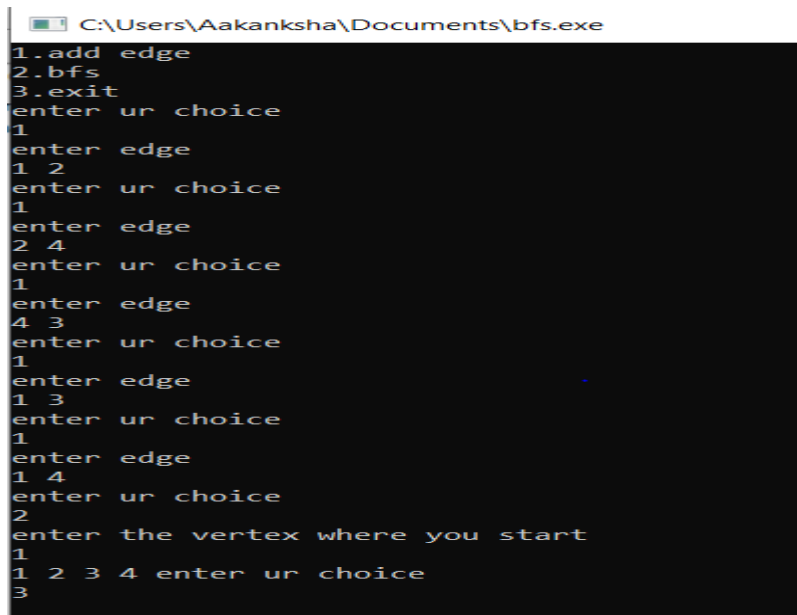
        while(!queue.empty()){
            s=queue.front();
            cout<<s<<" ";
            queue.pop_front();
            for(i=adj[s].begin();i!=adj[s].end();i++){
                if(!visited[*i]){
                    visited[*i]=true;
                    queue.push_back(*i);
                }
            }
        }
    }

int main(){
    graph obj(10);
    int v,e,ch;
    cout<<"1.add edge"<<endl;
    cout<<"2.bfs"<<endl;
    cout<<"3.exit"<<endl;
    while(1){
        cout<<"enter ur choice"<<endl;
        cin>>ch;
        switch(ch){
            case 1:cout<<"enter edge"<<endl;
                cin>>v>>e;
                obj.addedge(v,e);
                break;
            case 2:
                cout<<"enter the vertex where you start"<<endl;
                cin>>v;
                obj.bfs(v);
                break;
            case 3:

```

```
        exit(1);  
        break;  
    }  
}
```

output:



```
C:\Users\Aakanksha\Documents\bfs.exe  
1.add edge  
2.bfs  
3.exit  
enter ur choice  
1  
enter edge  
1 2  
enter ur choice  
1  
enter edge  
2 4  
enter ur choice  
1  
enter edge  
4 3  
enter ur choice  
1  
enter edge  
1 3  
enter ur choice  
1  
enter edge  
1 4  
enter ur choice  
2  
enter the vertex where you start  
1  
1 2 3 4 enter ur choice  
3
```

Problem statement:

Write a c++ program to traverse a given graph using DFS algorithm.

Description: Depth first traversal is a graph traversal algorithm that starts traversing the graph from root node and explores as far as possible along each branch before backtracking.

Program:

```
#include <iostream>

#include<conio.h>

#include<list>

using namespace std;

class graph {
int V;

list<int>* adj;

bool* visited;

public:

graph(int V);

void addedge(int v, int w);

void dfs(int v);

};

graph::graph(int V) {
this->V = V;

adj = new list<int>[V];

visited = new bool[V];

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

visited[i] = false;

}

void graph::addedge(int v, int w) {

adj[v].push_back(w);

}

void graph::dfs(int v){

visited[v] = true;
```

```

cout << v << " ";
list<int>::iterator i;
for (i = adj[v].begin(); i != adj[v].end(); ++i)
if (!visited[*i])
dfs(*i);
}
int main(){
    graph obj(10);
    int v,e,ch;
    cout<<"1.add edge"<<endl;
    cout<<"2.dfs"<<endl;
    cout<<"3.exit"<<endl;
    while(1){
cout<<"enter ur choice"<<endl;
cin>>ch;
switch(ch){
case 1:cout<<"enter edge"<<endl;
cin>>v>>e;
obj.addedge(v,e);
break;
case 2:
    cout<<"enter the vertex where you start"<<endl;
    cin>>v;
    cout<<"dfs:"<<endl;
obj.dfs(v);
break;
case 3: exit(1);
    break;
}}
}

```

Output:

```
C:\Users\Aakanksha\Documents\dfs.exe
1.add edge
2.dfs
3.exit
enter ur choice
1
enter edge
1 2
enter ur choice
1
enter edge
2 3
enter ur choice
1
enter edge
3 3
enter ur choice
1
enter edge
2 0
enter ur choice
1
enter edge
0 1
enter ur choice
1
enter edge
0 2
enter ur choice
2
enter the vertex where you start
2
dfs:
2 3 0 1 enter ur choice
```