**1.Write a C++ program to perform the following operations :**

**a) Create a Singly linked list of elements.**

**b) Search for a given element in the above list.**

**c) Delete an element from the above Singly linked list**

**d) Display the contents of the list**

#include<iostream.h>

#include<conio.h>

struct node

{

int data ;

node \*link ;

};

class singlelink

{

private :

node \* head;

public :

node \*head ;

public :

single link ()

{

head=new node;

head->data =null;

head->link=null;

void insert-front(int);

void insert-last(int);

void delete-front();

void delete-end();

void display();

void empty();

};

void singlelink:insert front (int n)

{

node\*temp;

temp=new node;

temp->data=n;

temp->link=null;

if(head->data==null&&head->link==null)

head=temp;

else

{

temp->link=head;

head=temp;

}

}

void singlelink::insert last (int m)

{

node \*temp=new node;

temp->data=m;

temp->link=null;

if(head->data ==null &&head->lint==null)

head=temp;

node(=\*prev;

prev=head;

while(prev->link!=null)

{

prev=prev->link;

}

prev->link=temp;

}

void singlelink :: insert -mid(int pos,int n)

{

node \*temp=new node;

temp->data=n;

temp->link=null;

if (head ->data ==null&&head ->link =null)

{

cout<<"ur list is empty so possible to insert at first position:";

head=temp;

}

else

{

int i=1;

node \*temp;

node \*next=head;

while((<=(pos-1))

{

temp1=next->link;

i++;

}

temp->link=next;

temp1->link =temp;

}}

void single link :: delte front ()

{

node \*temp;

temp=head;

if(head->data ==null && head->link==null)

cout<<"your list is empty ";

else

{

node \*t;

t=head;

temp=temp->link;

head=temp;

delete t;

}

}

void singler link ::delete-end()

{

node \*temp;

temp=head;

if (head ->datA ==null && head->link==null)

cout<<"your list is empty";

else

{

node\*temp1=head;

while(temp1->link!=null)

{

temp1=temp1->link;

}

delete temp;

}

void singalelink::delete val (int n)

{

node \*temp;

temp=head;

if(head->data==NULL && head->link== null)

cout<<"your list is empty";

else

{

node \*prev=head;

node \*temp1=head;

while(temp1->data!=m)

{

prev=temp1;

temp1=temp->link;

}

prev->link=temp->link;

delete temp1;

}}

void single link::search(int s)

{

if (head->data==null && head->link==null)

cout<<"your list is empty";

else

{

int i=0;

node \*temp1=head;

while(temp1->link!=null)

{

if(temp1->data==S)

{

i=1;

cout<<"found";

break;

}

else

temp=temp->link;

}

else

temp=temp->link;

}

if(i==0)

cout<<"not found ";

}

}

void singlink == empty()

{

node \*temp;

if (head ->data==null && head->link==null)

cout<<"list is not empty ";

}

void singelink :: disp()

{

node \*temp;

temp=head;

if head->data==null && temp->link==null)

cout<<"your list is empty ";

else

{

while (temp->link!=null)

{

cout << temp->data<<"->";

temp=temp->link;

}

cout << temp->data;

}

}

void single ::~single link()

{

node \*temp;

temp=head;

if temp->data==null&& temp->link==null)

delete temp;

else

{

while (temp!=null)

{

t=temp;

temp=temp->link;

delete t;

}

}

}

void main ()

{

int ch;

cout <<"enter your choice ...";

cin>>ch;

single link sl;

switch (ch)

{

case 1:cout <<"insert vthe front values ";

int a[10],i;

for (i=0;i<=3;i++)

{

cin >>a[i];

sl.display ();

break;

case 2: cout <<"inserting values out last :";

int b[10],5;

for (j=0;j<=3;j++)

{

cin >> b[j];

sl.insert last (b[j]);

}

sl.disp();

break ;

case 3: cout <<"delete value at front ";

sl.insert front [5];

sl.insert front[7];

sl.disp();

sl.delete front ();

sl.disp();

break;

case4 :cout<<"delete value at end ";

sl.insert -front (100);

sl.insert-front (200);

sl.disp();

break;

case 6 :cout <<"terminate the program";

break;

getch ();

}

**out put :**

save : single lik.cpp

ctrl+f9

enter your chocies

1.insert front

2.insert last

3.delete front

4.delete end

5.terminate program

insert the front values

10 20 30 40

40

30

20

10

enter your choice:5

30

20

10

enter your choice :6

terminating program.

**2. Write a C++ program to perform the following operations:  
a) Create a doubly linked list of elements.  
b) Delete an element from the above doubly linked list  
c) Display the contents of the list**.

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

struct node

{

int info;

struct node \*next;

struct node \*prev;

}\*start;

class double\_llist

{

public:

void create\_list(int value);

void add\_begin(int value);

void add\_after(int value, int position);

void delete\_element(int value);

void search\_element(int value);

void display\_dlist();

void count();

void reverse();

double\_llist()

{

start = NULL;

}

};

int main()

{

int choice, element, position;

double\_llist dl;

while (1)

{

cout<<endl<<"----------------------------"<<endl;

cout<<endl<<"Operations on Doubly linked list"<<endl;

cout<<endl<<"----------------------------"<<endl;

cout<<"1.Create Node"<<endl;

cout<<"2.Add at begining"<<endl;

cout<<"3.Add after position"<<endl;

cout<<"4.Delete"<<endl;

cout<<"5.Display"<<endl;

cout<<"6.Count"<<endl;

cout<<"7.Reverse"<<endl;

cout<<"8.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch ( choice )

{

case 1:

cout<<"Enter the element: ";

cin>>element;

dl.create\_list(element);

cout<<endl;

break;

case 2:

cout<<"Enter the element: ";

cin>>element;

dl.add\_begin(element);

cout<<endl;

break;

case 3:

cout<<"Enter the element: ";

cin>>element;

cout<<"Insert Element after postion: ";

cin>>position;

dl.add\_after(element, position);

cout<<endl;

break;

case 4:

if (start == NULL)

{

cout<<"List empty,nothing to delete"<<endl;

break;

}

cout<<"Enter the element for deletion: ";

cin>>element;

dl.delete\_element(element);

cout<<endl;

break;

case 5:

dl.display\_dlist();

cout<<endl;

break;

case 6:

exit(1);

default:

cout<<"Wrong choice"<<endl;

}

}

return 0;

}

void double\_llist::create\_list(int value)

{

struct node \*s, \*temp;

temp = new(struct node);

temp->info = value;

temp->next = NULL;

if (start == NULL)

{

temp->prev = NULL;

start = temp;

}

else

{

s = start;

while (s->next != NULL)

s = s->next;

s->next = temp;

temp->prev = s;

}

}

void double\_llist::add\_begin(int value)

{

if (start == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp;

temp = new(struct node);

temp->prev = NULL;

temp->info = value;

temp->next = start;

start->prev = temp;

start = temp;

cout<<"Element Inserted"<<endl;

}

void double\_llist::add\_after(int value, int pos)

{

if (start == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*tmp, \*q;

int i;

q = start;

for (i = 0;i < pos - 1;i++)

{

q = q->next;

if (q == NULL)

{

cout<<"There are less than ";

cout<<pos<<" elements."<<endl;

return;

}

}

tmp = new(struct node);

tmp->info = value;

if (q->next == NULL)

{

q->next = tmp;

tmp->next = NULL;

tmp->prev = q;

}

else

{

tmp->next = q->next;

tmp->next->prev = tmp;

q->next = tmp;

tmp->prev = q;

}

cout<<"Element Inserted"<<endl;

}

void double\_llist::delete\_element(int value)

{

struct node \*tmp, \*q;

if (start->info == value)

{

tmp = start;

start = start->next;

start->prev = NULL;

cout<<"Element Deleted"<<endl;

free(tmp);

return;

}

q = start;

while (q->next->next != NULL)

{

if (q->next->info == value)

{

tmp = q->next;

q->next = tmp->next;

tmp->next->prev = q;

cout<<"Element Deleted"<<endl;

free(tmp);

return;

}

q = q->next;

}

if (q->next->info == value)

{

tmp = q->next;

free(tmp);

q->next = NULL;

cout<<"Element Deleted"<<endl;

return;

}

cout<<"Element "<<value<<" not found"<<endl;

}

void double\_llist::display\_dlist()

{

struct node \*q;

if (start == NULL)

{

cout<<"List empty,nothing to display"<<endl;

return;

}

q = start;

cout<<"The Doubly Link List is :"<<endl;

while (q != NULL)

{

cout<<q->info<<" <-> ";

q = q->next;

}

cout<<"NULL"<<endl;

}

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 2

Enter the element: 100

First Create the list.

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 3

Enter the element: 200

Insert Element after postion: 1

First Create the list.

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 4

List empty,nothing to delete

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 5

List empty,nothing to display

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 1

Enter the element: 100

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 5

The Doubly Link List is :

100 <-> NULL

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 2

Enter the element: 200

Element Inserted

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 5

The Doubly Link List is :

200 <-> 100 <-> NULL

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 3

Enter the element: 50

Insert Element after postion: 2

Element Inserted

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 5

The Doubly Link List is :

200 <-> 100 <-> 50 <-> NULL

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 3

Enter the element: 150

Insert Element after postion: 3

Element Inserted

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 5

The Doubly Link List is :

200 <-> 100 <-> 50 <-> 150 <-> NULL

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 6

Number of elements are: 4

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 4

Enter the element for deletion: 50

Element Deleted

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Quit

Enter your choice : 5

The Doubly Link List is :

200 <-> 100 <-> 150 <-> NULL

**3.Write C++ programs to implement the following using an array**

**a) stack with ADT in array**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class stack

{

int stk[5];

int top;

public:

stack()

{

top=-1;

}

void push(int x)

{

if(top > 4)

{

cout <<"stack over flow";

return;

}

stk[++top]=x;

cout <<"inserted" <<x;

}

void pop()

{

if(top <0)

{

cout <<"stack under flow";

return;

}

cout <<"deleted" <<stk[top--];

}

void display()

{

if(top<0)

{

cout <<" stack empty";

return;

}

for(int i=top;i>=0;i--)

cout <<stk[i] <<" ";

}};

main()

{

int ch;

stack st;

while(1)

{

cout <<"\n1.push 2.pop 3.display 4.exit\nEnter ur choice";

cin >> ch;

switch(ch)

{

case 1: cout <<"enter the element";

cin >> ch;

st.push(ch);

break;

case 2: st.pop(); break;

case 3: st.display();break;

case 4: exit(0);

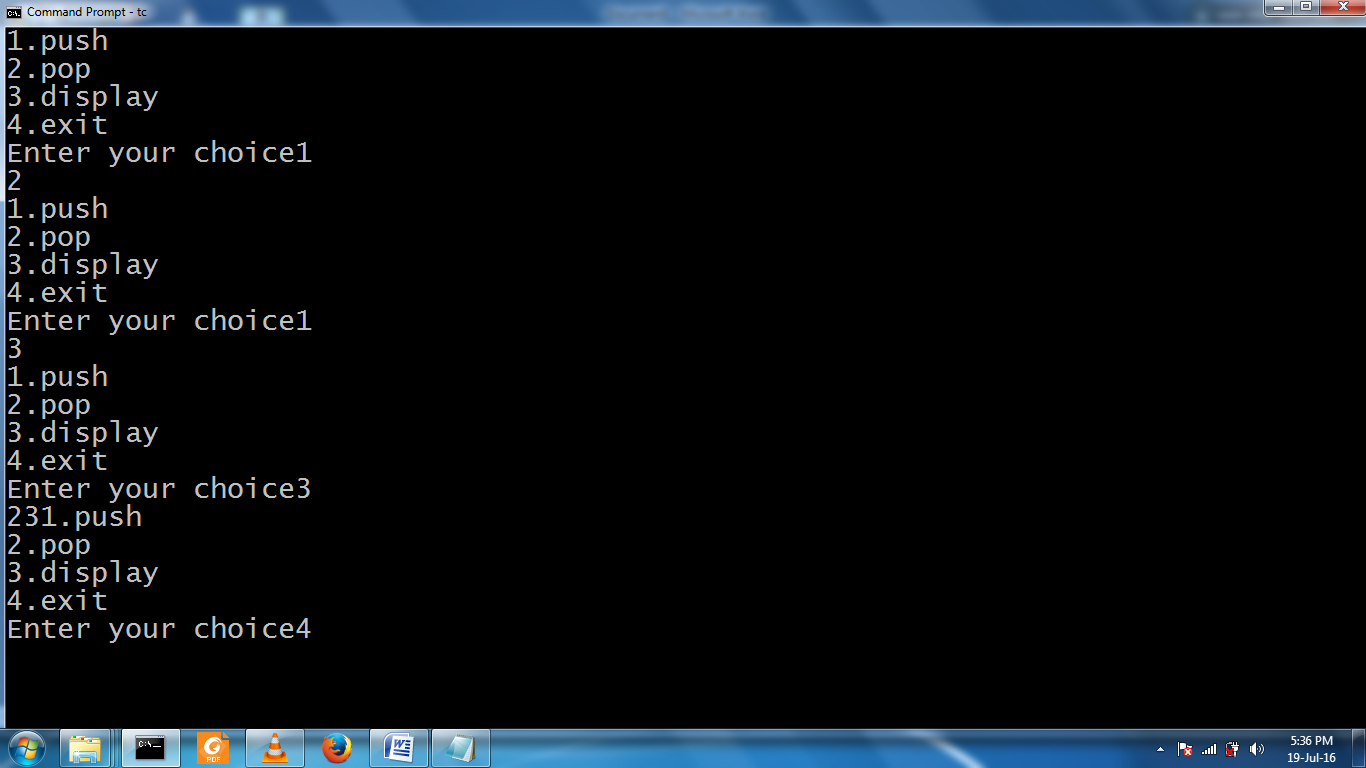
}

}

return (0);

}

Input/output



**b.) Queue ADT in array**

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class queue

{

int queue1[5];

int rear,front;

public:

queue()

{

rear=-1;

front=-1;

}

void insert(int x)

{

if(rear > 4)

{

cout <<"queue over flow";

front=rear=-1;

return;

}

queue1[++rear]=x;

cout <<"inserted" <<x;

}

void delet()

{

if(front==rear)

{

cout <<"queue under flow";

return;

}

cout <<"deleted" <<queue1[++front];

}

void display()

{

if(rear==front)

{

cout <<" queue empty";

return;

}

for(int i=front+1;i<=rear;i++)

cout <<queue1[i]<<" ";

}

};

main()

{

int ch;

queue qu;

while(1)

{

cout <<"\n1.insert 2.delet 3.display 4.exit\nEnter ur choice";

cin >> ch;

switch(ch)

{

case 1: cout <<"enter the element";

cin >> ch;

qu.insert(ch);

break;

case 2: qu.delet(); break;

case 3: qu.display();break;

case 4: exit(0);

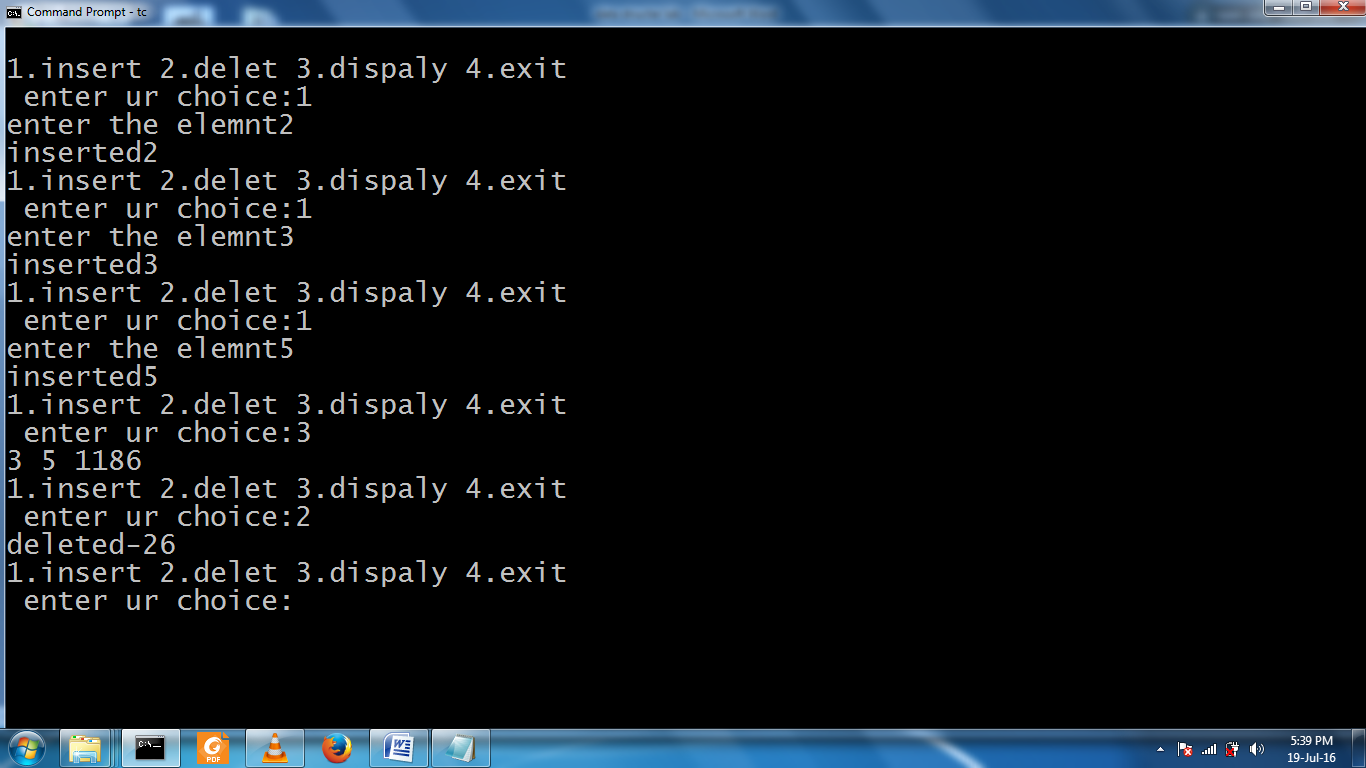
}

}

return (0);

}

Input/output



**4.Write C++ programs to implement the following using a singly linked list.**

**a) linked list using stack adt**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class node

{

public:

class node \*next;

int data;

};

class stack : public node

{

node \*head;

int tos;

public:

stack()

{

tos=-1;

}

void push(int x)

{

if (tos < 0 )

{

head =new node;

head->next=NULL;

head->data=x;

tos ++;

}

else

{

node \*temp,\*temp1;

temp=head;

if(tos >= 4)

{

cout <<"stack over flow";

return;

}

tos++;

while(temp->next != NULL)

temp=temp->next;

temp1=new node;

temp->next=temp1;

temp1->next=NULL;

temp1->data=x;

}

}

void display()

{

node \*temp;

temp=head;

if (tos < 0)

{

cout <<" stack under flow";

return;

}

while(temp != NULL)

{

cout <<temp->data<< " ";

temp=temp->next;

}

}

void pop()

{

node \*temp;

temp=head;

if( tos < 0 )

{

cout <<"stack under flow";

return;

}

tos--;

while(temp->next->next!=NULL)

{

temp=temp->next;

}

temp->next=NULL;

}

};

main()

{

stack s1;

int ch;

while(1)

{

cout <<"\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n enter ur choice:";

cin >> ch;

switch(ch)

{

case 1: cout <<"\n enter a element";

cin >> ch;

s1.push(ch);

break;

case 2: s1.pop();break;

case 3: s1.display();

break;

case 4: exit(0);

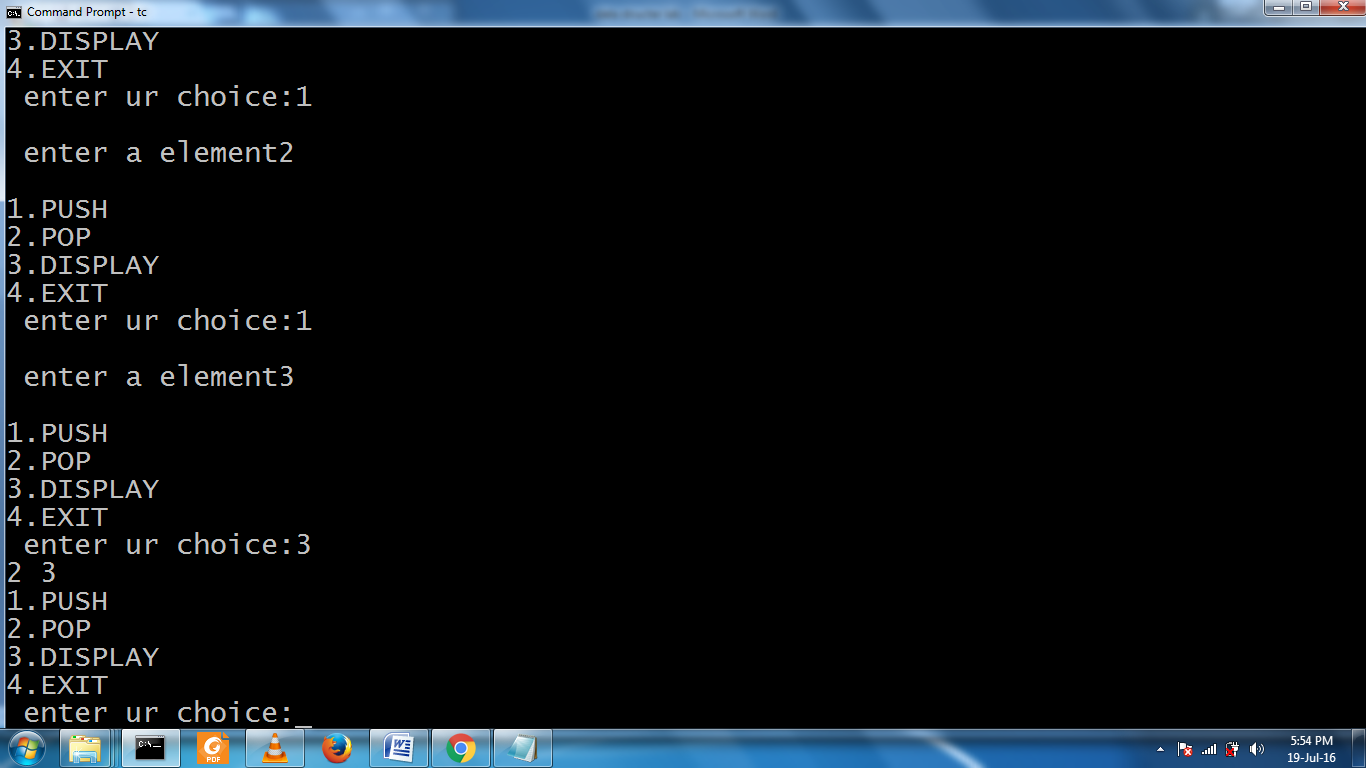
}

}

return (0);

}

Input/ouput



**b> queue ADT link list**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class node

{

public:

class node \*next;q

int data;

};

class queue : public node

{

node \*head;

int front,rare;

public:

queue()

{

front=-1;

rare=-1;

}

void push(int x)

{

if (rare < 0 )

{

head =new node;

head->next=NULL;

head->data=x;

rare ++;

}

else

{

node \*temp,\*temp1;

temp=head;

if(rare >= 4)

{

cout <<"queue over flow";

return;

}

rare++;

while(temp->next != NULL)

temp=temp->next;

temp1=new node;

temp->next=temp1;

temp1->next=NULL;

temp1->data=x;

} }

void display()

{

node \*temp;

temp=head;

if (rare < 0)

{

cout <<" queue under flow";

return;

}

while(temp != NULL)

{

cout <<temp->data<< " ";

temp=temp->next;

}

}

void pop()

{

node \*temp;

temp=head;

if( rare < 0)

{

cout <<"queue under flow";

return;

}

if(front == rare)

{

front = rare =-1;

head=NULL;

return;

}

front++;

head=head->next;

}

};

main()

{

queue s1;

int ch;

while(1)

{

cout<<"\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n enter ru choice:";

cin >> ch;

switch(ch)

{

case 1:

cout <<"\n enter a element";

cin >> ch;

s1.push(ch); break;

case 2: s1.pop();break;

case 3: s1.display();break;

case 4: exit(0);

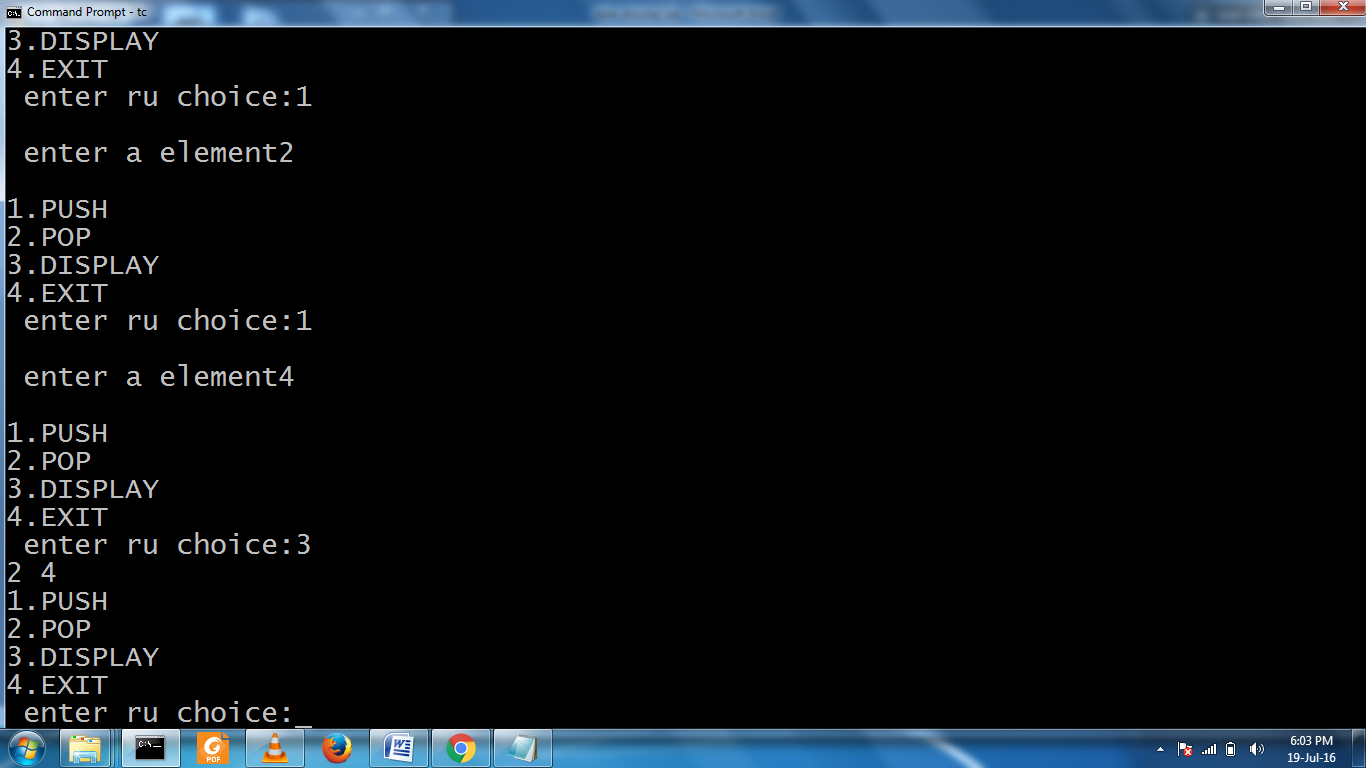
}

}

return (0);

}

Input/output



5. Write a C++ program to convert

i) a given infix expression into postfix form using stack

and ii) evaluate the Postfix expression using stack.

6.Write C++ programs to implement the deque (double ended queue) ADT using

a)Singly linked list b) Doubly linked list c) an array.

**7. Write C++ programs that use recursive functions to traverse the given binary tree in**

**a) Preorder b) Inorder and c) Postorder**

#include <iostream>

using namespace std;

// Node class

class Node {

int key;

Node\* left;

Node\* right;

public:

Node() { key=-1; left=NULL; right=NULL; };

void setKey(int aKey) { key = aKey; };

void setLeft(Node\* aLeft) { left = aLeft; };

void setRight(Node\* aRight) { right = aRight; };

int Key() { return key; };

Node\* Left() { return left; };

Node\* Right() { return right; };

};

// Tree class

class Tree {

Node\* root;

public:

Tree();

~Tree();

Node\* Root() { return root; };

void addNode(int key);

void inOrder(Node\* n);

void preOrder(Node\* n);

void postOrder(Node\* n);

private:

void addNode(int key, Node\* leaf);

void freeNode(Node\* leaf);

};

// Constructor

Tree::Tree() {

root = NULL;

}

// Destructor

Tree::~Tree() {

freeNode(root);

}

// Free the node

void Tree::freeNode(Node\* leaf)

{

if ( leaf != NULL )

{

freeNode(leaf->Left());

freeNode(leaf->Right());

delete leaf;

}

}

// Add a node

void Tree::addNode(int key) {

// No elements. Add the root

if ( root == NULL ) {

cout << "add root node ... " << key << endl;

Node\* n = new Node();

n->setKey(key);

root = n;

}

else {

cout << "add other node ... " << key << endl;

addNode(key, root);

}

}

// Add a node (private)

void Tree::addNode(int key, Node\* leaf) {

if ( key <= leaf->Key() ) {

if ( leaf->Left() != NULL )

addNode(key, leaf->Left());

else {

Node\* n = new Node();

n->setKey(key);

leaf->setLeft(n);

}

}

else {

if ( leaf->Right() != NULL )

addNode(key, leaf->Right());

else {

Node\* n = new Node();

n->setKey(key);

leaf->setRight(n);

}

}

}

// Print the tree in-order

// Traverse the left sub-tree, root, right sub-tree

void Tree::inOrder(Node\* n) {

if ( n ) {

inOrder(n->Left());

cout << n->Key() << " ";

inOrder(n->Right());

}

}

// Print the tree pre-order

// Traverse the root, left sub-tree, right sub-tree

void Tree::preOrder(Node\* n) {

if ( n ) {

cout << n->Key() << " ";

preOrder(n->Left());

preOrder(n->Right());

}

}

// Print the tree post-order

// Traverse left sub-tree, right sub-tree, root

void Tree::postOrder(Node\* n) {

if ( n ) {

postOrder(n->Left());

postOrder(n->Right());

**cout << n->Key() << " ";**

}

}

// Test main program

int main() {

Tree\* tree = new Tree();

tree->addNode(30);

tree->addNode(10);

tree->addNode(20);

tree->addNode(40);

tree->addNode(50);

cout << "In order traversal" << endl;

tree->inOrder(tree->Root());

cout << endl;

cout << "Pre order traversal" << endl;

tree->preOrder(tree->Root());

cout << endl;

cout << "Post order traversal" << endl;

tree->postOrder(tree->Root());

cout << endl;

delete tree;

return 0;

}

.

**OUTPUT:-**

add root node ... 30

add other node ... 10

add other node ... 20

add other node ... 40

add other node ... 50

In order traversal

10 20 30 40 50

Pre order traversal

30 10 20 40 50

Post order traversal

20 10 50 40 30

8.Write a C++ program to perform the following operations:

a) Construct a binary search tree of elements.

b) Search for a key element in the above binary search tree.

c) Delete an element from the above binary search tree.

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

void insert(int,int );

void delte(int);

void display(int);

int search(int);

int search1(int,int);

int tree[40],t=1,s,x,i;

main()

{

int ch,y;

for(i=1;i<40;i++)

tree[i]=-1;

while(1)

{

cout <<"1.INSERT\n2.DELETE\n3.DISPLAY\n4.SEARCH\n5.EXIT\nEnter your choice:";

cin >> ch;

switch(ch)

{

case 1:

cout <<"enter the element to insert";

cin >> ch;

insert(1,ch);

break;

case 2:

cout <<"enter the element to delete";

cin >>x;

y=search(1);

if(y!=-1) delte(y);

else cout<<"no such element in tree";

break;

case 3:

display(1);

cout<<"\n";

for(int i=0;i<=32;i++)

cout <<i;

cout <<"\n";

break;

case 4:

cout <<"enter the element to search:";

cin >> x;

y=search(1);

if(y == -1) cout <<"no such element in tree";

else cout <<x << "is in" <<y <<"position";

break;

case 5:

exit(0);

}

}

}

void insert(int s,int ch )

{

int x;

if(t==1)

{

tree[t++]=ch;

return;

}

x=search1(s,ch);

if(tree[x]>ch)

tree[2\*x]=ch;

else

tree[2\*x+1]=ch;

t++;

}

void delte(int x)

{

if( tree[2\*x]==-1 && tree[2\*x+1]==-1)

tree[x]=-1;

else if(tree[2\*x]==-1)

{ tree[x]=tree[2\*x+1];

tree[2\*x+1]=-1;

}

else if(tree[2\*x+1]==-1)

{ tree[x]=tree[2\*x];

tree[2\*x]=-1;

}

else

{

tree[x]=tree[2\*x];

delte(2\*x);

}

t--;

}

int search(int s)

{

if(t==1)

{

cout <<"no element in tree";

return -1;

}

if(tree[s]==-1)

return tree[s];

if(tree[s]>x)

search(2\*s);

else if(tree[s]<x)

search(2\*s+1);

else

return s;

}

void display(int s)

{

if(t==1)

{cout <<"no element in tree:";

return;}

for(int i=1;i<40;i++)

if(tree[i]==-1)

cout <<" ";

else cout <<tree[i];

return ;

}

int search1(int s,int ch)

{

if(t==1)

{

cout <<"no element in tree";

return -1;

}

if(tree[s]==-1)

return s/2;

if(tree[s] > ch)

search1(2\*s,ch);

else search1(2\*s+1,ch);

}

**OUTPUT**  
1.INSERT  
2.DELETE  
3.DISPLAY  
4.SEARCH  
5.EXIT  
Enter your choice:3

no element in tree:  
0123456789011121314151617181920212223242526272829303132

1.INSERT  
2.DELETE  
3.DISPLAY  
4.SEARCH  
5.EXIT  
Enter your choice:1

Enter the element to insert 10  
1.INSERT  
2.DELETE  
3.DISPLAY  
4.SEARCH  
5.EXIT  
Enter your choice:4

Enter the element to search: 10  
10 is in 1 position  
1.INSERT  
2.DELETE  
3.DISPLAY  
4.SEARCH  
5.EXIT

Enter your choice:5

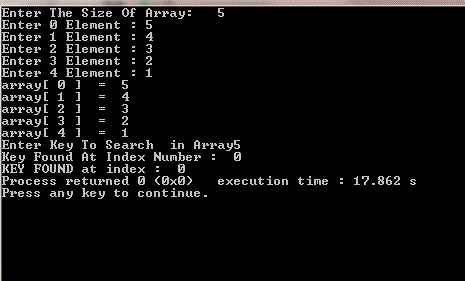
**9.Write C++ programs for implementing the following Searching methods:**

**a) Linear Search b) Binary Search**

**a> linear search**

#include<iostream>  
using namespace std;  
  
int main() {  
cout<<"Enter The Size Of Array:   ";  
int size;  
cin>>size;  
  
  
int array[size], key,i;  
  
// Taking Input In Array  
 for(int j=0;j<size;j++){  
 cout<<"Enter "<<j<<" Element: ";  
 cin>>array[j];  
 }  
  
//Your Entered Array Is  
 for(int a=0;a<size;a++){  
    cout<<"array[ "<<a<<" ]  =  ";  
    cout<<array[a]<<endl;  
 }  
  
 cout<<"Enter Key To Search  in Array";  
 cin>>key;  
  
   for(i=0;i<size;i++){  
    if(key==array[i]){  
 cout<<"Key Found At Index Number :  "<<i<<endl;  
 break;  
    }  
 }  
  
  
if(i != size){  
cout<<"KEY FOUND at index :  "<<i;  
}  
else{  
cout<<"KEY NOT FOUND in Array  ";  
}  
   return 0;  
}

**Output**



**b> binary search**

#include<stdio.h>

#include<iostream.h>

#include<conio.h>

void main(){

int a[10],num,n,beg,end,mid ,f=0,i;

clrscr();

cout<<"\n enter the number of elements in the array : ";

cin>>n;

cout<<"\n enter the element :";

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

cin>>a[i];

}

cout<<"\n eneter the number thst has to be searched :";

cin>>num;

beg=0;

end=n-1;

while(beg<=end){

mid=(beg+end)/2;

if(a[mid]==num){

cout<<"\n is present in the array at position ="<<num<<mid;

f=1;

break;

}

else if(a[mid]>num)

end=mid-1;

else

end=mid+1;

}

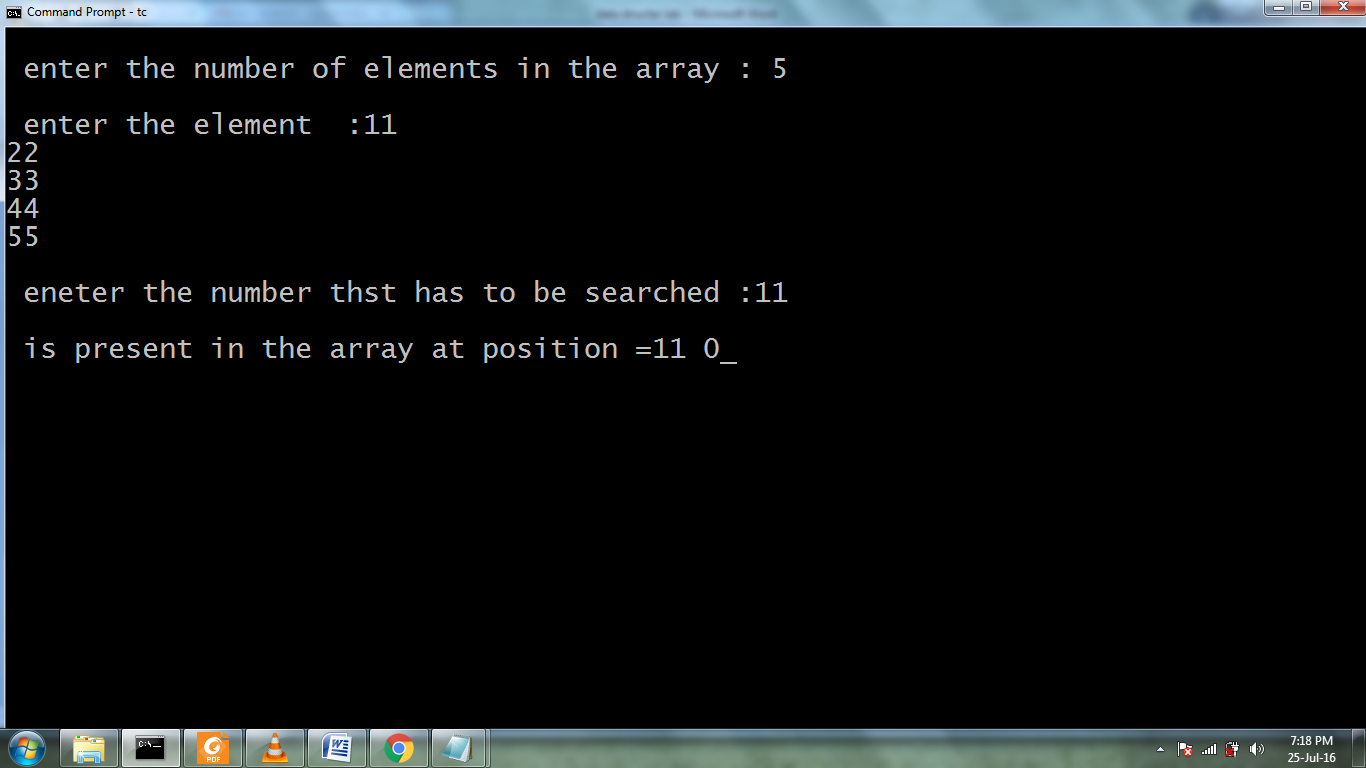
if(beg>end && f==0)

cout<<"does not exit in the array "<<num;

getch();

}

**Output**



**10.Write C++ programs for implementing the following sorting methods:**

**a) Bubble Sort b) Selection Sort c) Insertion Sort**

**a) Bubble Sort**

#include<iostream>

using namespace std;

void main()

{

int a[50],i,j,n;

printf("Enter array elements\n");

scanf("%d",&n);

printf("Array Elements\n");

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

scanf("%d",&a[i]);

for(i=0;i<n-1;i++)

{

for(j=0;j<n-1;j++)

{

if(a[j]>a[j+1])

{

int temp;

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("sorted array(bubble sort)\n");

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

printf("%5d",a[i]);

}

**Output**

Enter array elements 5

99 55 77 22 11 44

Array Elements

99 55 77 22 11 44

sorted array(bubble sort)

11 22 44 55 77 99

**b) Selection Sort**

#include<iostream>

 using namespace std;

 int main()

{

    int i,j,n,loc,temp,min,a[30];

    cout<<"Enter the number of elements:";

    cin>>n;

    cout<<"\nEnter the elements\n";

    for(i=0;i<n;i++)

    {

        cin>>a[i];

    }

     for(i=0;i<n-1;i++)

    {

        min=a[i];

        loc=i;

        for(j=i+1;j<n;j++)

        {

            if(min>a[j])

            {

                min=a[j];

                loc=j;

            }

        }

         temp=a[i];

        a[i]=a[loc];

        a[loc]=temp;

    }

    cout<<"\nSorted list is as follows\n";

    for(i=0;i<n;i++)

    {

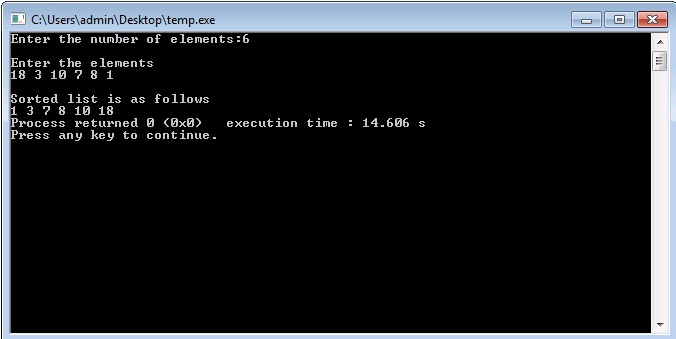
        cout<<a[i]<<" ";

    }

    return 0;

}

**Output:-**

****

**c) Insertion sorting**

#include<iostream>

using namespace std;

void INSERTION\_SORT(int [],int );

int main()

{

char ch;

do

{

int n;

cout<<"Enter the no. of elements in the array"<<endl;

cin>>n;

int a[n];

cout<<"Enter the elements in the array"<<endl;

for(int i=1;i<=n;i++)

{

cin>>a[i];

}

INSERTION\_SORT(a,n);

cout<<"::::Sorted Output:::"<<endl;

for(int k=1;k<=n;k++)

{

cout<<a[k]<<" ";

}

cout<<endl<<"Press 'y' or 'Y' to continue!!!";

cin>>ch;

}while(ch=='y'||ch=='Y');

return 0;

}

void INSERTION\_SORT(int a[],int n)

{

int key,i,j;

for(j=2;j<=n;j++)

{

key=a[j];

i=j-1;

while((i>0)&&(a[i]>key))

{

a[i+1]=a[i];

i=i-1;

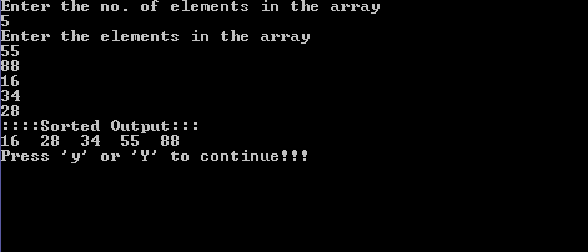
}

a[i+1]=key;

}

}

Output



**11.Write C++ programs for implementing the following sorting methods:**

**a)Merge sort b)Quick sort c) Heap sort d)Radix sort.**

**a) /\* C++ Program to implement Merge Sort \*/**

#include<iostream>

using namespace std;

void merge\_sort(int [],int ,int );

void merge(int [],int,int ,int );

int main()

{

int n;

cout<<"Enter the size of the array"<<endl;

cin>>n;

int a[n];

cout<<"Enter the elements in the array"<<endl;

for(int i=1;i<=n;i++)

{

cin>>a[i];

}

cout<<"sorting using merge sort"<<endl;

int p=1,r=n;

merge\_sort(a,p,r);

cout<<"sorted form"<<endl;

for(int i=1;i<=n;i++)

{

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

}

return 0;

}

void merge\_sort(int a[],int p,int r)

{

int q;

if(p<r)

{

q=(p+r)/2;

merge\_sort(a,p,q);

merge\_sort(a,q+1,r);

merge(a,p,q,r);

}

}

void merge(int a[],int p,int q,int r)

{

cout<<"Entered merge"<<endl;

int n1=q-p+1;

int n2=r-q;

int L[n1+1];

int R[n2+1];

for(int i=1;i<=n1;i++)

{

L[i]=a[p+i-1];

}

for(int j=1;j<=n2;j++)

{

R[j]=a[q+j];

}

L[n1+1]=999;

R[n2+1]=999;

int i=1, j=1;

for(int k=p;k<=r;k++)

{

if(L[i]<=R[j])

{

a[k]=L[i];

i=i+1;

}

else

{

a[k]=R[j];

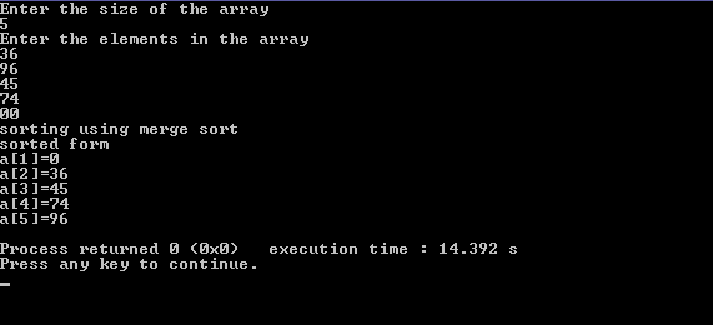
j=j+1;

}

}

}

**//Output of the above program**



**b>quick sort**

#include<stdio.h>

#include<conio.h>

#include<iostream.h>

void quicksort(int a[],int lb,int ub)

{

int key=lb,i=lb,j=ub;

printf("\ni=%d j=%d key=%d\n",i,j,key);

while(i<j)

{

while(a[i]<=a[key])

i++;

while(a[j]>a[key])

j--;

if(i<j)

a[i]=((a[i]+a[j])-(a[i]=a[j]));

a[key]=((a[key]+a[j])-(a[j]=a[key]));

quicksort(a,lb,j-1);

quicksort(a,j+1,ub);

}

}

void main()

{

int i,a[30],n;

clrscr();

cout<<"enter the how many no";

cin>>n;

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

cin>>a[i];

cout<<"before elemnet\n"<<" ";

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

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

quicksort(a,0,n-1);

cout<<"\nafetr sorting"<<" ";

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

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

getch();

}

**Output**

enter the size of list 5

enter the elements 99

88

77

66

55

after sorting element : 55 66 77 88 99

**c>heap sort**

#include<stdio.h>

#include<conio.h>

#include<iostream.h>

void heapsort(int [ ],int);

void buildheap(int [ ],int);

void satisfyheap(int[ ],int,int);

void main(){

int a[10],i,s;

clrscr();

cout<<"enetr the size of list ";

cin>>s;

cout<<"enetr the elements";

for(i=0;i<s;i++){

cin>>a[i];

}

heapsort(a,s);

getch();

}

void heapsort(int a[ ],int l){

buildheap(a,l);

int h,i,temp;

h=l-1;

for(i=h;i>=0;i--){

temp=a[0];

a[0]=a[h];

a[h]=temp;

h--;

satisfyheap(a,0,h);

}

cout<<"after sorting element :";

for(i=0;i<l;i++){

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

}

}

void buildheap(int a[ ],int l){

int i,h;

h=l-1;

for(i=(l/2);i>=0;i--){

satisfyheap(a,i,h);

}

}

void satisfyheap(int a[ ],int i,int h){

int l1,l,r,temp;

l1=2\*i;

r=2\*i+1;

if(l1<=h && a[l1]>a[i]){

l=l1;

}

else{

l=i;

}

if(r<=h && a[r]>a[l]){

l=r;

}

if(l!=i){

temp=a[i];

a[i]=a[l];

a[l]=temp;

satisfyheap(a,l,h);

}

}

**Output**

enter the size of list

5

enter the elements99

88

77

66

55

after sorting element : 55 66 77 88 99

**d> radix sort**

#include<stdio.h>

#include<conio.h>

#include<iostream.h>

#include<math.h>

void radixsort(int a[ ] ,int m,int n){

int temp,i,j,k,y,d,b[10],p,q;

y=-1;

for(i=1;i<=m;i++){

q=1;

y++;

p=pow(10,y);

for(j=0;j<=9;j++){

for(k=1;k<=n;k++){

d=(a[k]/p)%10;

if(d==j)

b[q++]=a[k];

}

}

for(temp=1;temp<=n;temp++)

a[temp]=b[temp];

}

}

void main(){

int i,a[30],n,max,max1;

clrscr();

cout<<"enetr tehe size";

cin>>n;

cout<<"enetr the elements ";

for(i=1;i<=n;i++)

cin>>a[i];

cout<<"before\n";

for(i=1;i<=n;i++)

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

max=0;

for(i=1;i<=n;i++)

if(a[i]>max)

max=a[i];

max1=0;

while(max!=0){

max=max/10;

max1++;

}

radixsort(a,max1,n);

cout<<"\nafter\n";

for(i=1;i<=n;i++){

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

}

getch();

}

**Output**

enter the size of list 5

enter the elements99

88

77

66

55

after sorting element : 55 66 77 88 99

**12.Write a C++ program to perform the following operation:**

**a) Insertion into a B-tree**

#include<stdio.h>

#include<conio.h>

#include<iostream>

using namespace std;

struct BTreeNode

{

int \*data;

BTreeNode \*\*child\_ptr;

bool leaf;

int n;

}\*root = NULL, \*np = NULL, \*x = NULL;

BTreeNode \* init()

{

int i;

np = new BTreeNode;

np->data = new int[5];

np->child\_ptr = new BTreeNode \*[6];

np->leaf = true;

np->n = 0;

for (i = 0; i < 6; i++)

{

np->child\_ptr[i] = NULL;

}

return np;

. }

void traverse(BTreeNode \*p)

{

cout<<endl;

int i;

for (i = 0; i < p->n; i++)

{

if (p->leaf == false)

{

traverse(p->child\_ptr[i]);

}

cout << " " << p->data[i];

}

if (p->leaf == false)

{

traverse(p->child\_ptr[i]);

}

cout<<endl;

}

void sort(int \*p, int n)

{

int i, j, temp;

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

{

for (j = i; j <= n; j++)

{

if (p[i] > p[j])

{

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}

int split\_child(BTreeNode \*x, int i)

{

int j, mid;

BTreeNode \*np1, \*np3, \*y;

np3 = init();

np3->leaf = true;

if (i == -1)

{

mid = x->data[2];

x->data[2] = 0;

x->n--;

np1 = init();

np1->leaf = false;

x->leaf = true;

for (j = 3; j < 5; j++)

{

np3->data[j - 3] = x->data[j];

np3->child\_ptr[j - 3] = x->child\_ptr[j];

np3->n++;

x->data[j] = 0;

x->n--;

}

for (j = 0; j < 6; j++)

{

x->child\_ptr[j] = NULL;

}

np1->data[0] = mid;

np1->child\_ptr[np1->n] = x;

np1->child\_ptr[np1->n + 1] = np3;

np1->n++;

root = np1;

}

else

{

y = x->child\_ptr[i];

mid = y->data[2];

y->data[2] = 0;

y->n--;

for (j = 3; j < 5; j++)

{

np3->data[j - 3] = y->data[j];

np3->n++;

y->data[j] = 0;

y->n--;

}

x->child\_ptr[i + 1] = y;

x->child\_ptr[i + 1] = np3;

}

return mid;

}

void insert(int a)

{

int i, temp;

x = root;

if (x == NULL)

{

root = init();

x = root;

}

else

{

if (x->leaf == true && x->n == 5)

{

temp = split\_child(x, -1);

x = root;

for (i = 0; i < (x->n); i++)

{

if ((a > x->data[i]) && (a < x->data[i + 1]))

{

i++;

break;

}

else if (a < x->data[0])

{

break;

}

else

{

continue;

}

}

x = x->child\_ptr[i];

}

else

{

while (x->leaf == false)

{

for (i = 0; i < (x->n); i++)

{

if ((a > x->data[i]) && (a < x->data[i + 1]))

{

i++;

break;

}

else if (a < x->data[0])

{

break;

}

else

{

continue;

}

}

if ((x->child\_ptr[i])->n == 5)

{

temp = split\_child(x, i);

x->data[x->n] = temp;

x->n++;

continue;

}

else

{

x = x->child\_ptr[i];

}

}

}

}

x->data[x->n] = a;

sort(x->data, x->n);

x->n++;

}

int main()

{

int i, n, t;

cout<<"enter the no of elements to be inserted\n";

cin>>n;

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

{

cout<<"enter the element\n";

cin>>t;

insert(t);

}

cout<<"traversal of constructed tree\n";

traverse(root);

getch();

}

**Output**

enter the no of elements to be inserted

8

enter the element

10

enter the element

20

enter the element

5

enter the element

6

enter the element

12

enter the element

30

enter the element

7

enter the element

17

traversal of constructed tree

5 6 7

10

12 17 20 30

**13.Write a C++ program to implement all the functions of a dictionary (ADT) using hashing.**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

# define max 10

typedef struct list

{

int data;

struct list \*next;

}node\_type;

node\_type \*ptr[max],\*root[max],\*temp[max];

class Dictionary

{

public:

int index;

Dictionary();

void insert(int);

void search(int);

void delete\_ele(int);

};

Dictionary::Dictionary()

{

index=-1;

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

{

root[i]=NULL;

ptr[i]=NULL;

temp[i]=NULL;

}

}

void Dictionary::insert(int key)

{

index=int(key%max);

ptr[index]=(node\_type\*)malloc(sizeof(node\_type));

ptr[index]->data=key;

if(root[index]==NULL)

{

root[index]=ptr[index];

root[index]->next=NULL;

temp[index]=ptr[index];

}

else

{

temp[index]=root[index];

while(temp[index]->next!=NULL)

temp[index]=temp[index]->next;

temp[index]->next=ptr[index];

}

}

void Dictionary::search(int key)

{

int flag=0;

index=int(key%max);

temp[index]=root[index];

while(temp[index]!=NULL)

{

if(temp[index]->data==key)

{

cout<<"\nSearch key is found!!";

flag=1;

break;

}

else temp[index]=temp[index]->next;

}

if (flag==0)

cout<<"\nsearch key not found.......";

}

void Dictionary::delete\_ele(int key)

{

index=int(key%max);

temp[index]=root[index];

while(temp[index]->data!=key && temp[index]!=NULL)

{

ptr[index]=temp[index];

temp[index]=temp[index]->next;

}

ptr[index]->next=temp[index]->next;

cout<<"\n"<<temp[index]->data<<" has been deleted.";

temp[index]->data=-1;

temp[index]=NULL;

free(temp[index]);

}

main()

{

int val,ch,n,num;

char c;

Dictionary d;

do

{

cout<<"\nMENU:\n1.Create";

cout<<"\n2.Search for a value\n3.Delete an value";

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:cout<<"\nEnter the number of elements to be inserted:";

cin>>n;

cout<<"\nEnter the elements to be inserted:";

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

{

cin>>num;

d.insert(num);

}

break;

case 2:cout<<"\nEnter the element to be searched:";

cin>>n;

d.search(n);

case 3:cout<<"\nEnter the element to be deleted:";

cin>>n;

d.delete\_ele(n);

break;

default:cout<<"\nInvalid choice....";

}

cout<<"\nEnter y to continue......";

cin>>c;

}while(c=='y');

getch();

}

**Output**

1.Create  
2.Search for a value  
3.Delete an value  
Enter your choice:1

Enter the number of elements to be inserted:8

Enter the elements to be inserted:10 4 5 8 7 12 6 1

Enter y to continue……y

MENU:  
1.Create  
2.Search for a value  
3.Delete an value  
Enter your choice:2

Enter the element to be searched:12

Search key is found!!  
Enter the element to be deleted:1

1 has been deleted.  
Enter y to continue……y

**14.Write a C++ program for implementing Knuth-Morris- Pratt pattern matching algorithm.**

#include<iostream.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void computeLPSArray(char \*pat, int M, int \*lps);

void KMPSearch(char \*pat, char \*txt)

{

int M = strlen(pat);

int N = strlen(txt);

// create lps[] that will hold the longest prefix suffix values for pattern

int \*lps = (int \*)malloc(sizeof(int)\*M);

int j = 0; // index for pat[]

// Preprocess the pattern (calculate lps[] array)

computeLPSArray(pat, M, lps);

int i = 0; // index for txt[]

while(i < N)

{

if(pat[j] == txt[i])

{

j++;

i++;

}

if (j == M)

{

Cout<<"Found pattern at index %d \n"<< i-j;

j = lps[j-1];

}

else if(pat[j] != txt[i])

{

if(j != 0)

j = lps[j-1];

else

i = i+1;

}

}

free(lps);

}

void computeLPSArray(char \*pat, int M, int \*lps)

{

int len = 0; // lenght of the previous longest prefix suffix

int i;

lps[0] = 0; // lps[0] is always 0

i = 1;

// the loop calculates lps[i] for i = 1 to M-1

while(i < M)

{

if(pat[i] == pat[len])

{

len++;

lps[i] = len;

i++;

}

else // (pat[i] != pat[len])

{

if( len != 0 )

{

// This is tricky. Consider the example AAACAAAA and i = 7.

len = lps[len-1];

// Also, note that we do not increment i here

}

else // if (len == 0)

{

lps[i] = 0;

i++;

}

}

}

}

// Driver program to test above function

int main()

{

char \*txt = "ABABDABACDABABCABAB";

char \*pat = "ABABCABAB";

KMPSearch(pat, txt);

return 0;

}

**Output**

Found pattern at index 10

**15.Write C++ programs that use non-recursive functions to traverse the given**

**binary tree in**

**a) Preorder b) inorder and c) postorder.**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class node

{

public:

class node \*left;

class node \*right;

int data;

};

class tree: public node

{

public:

int stk[50],top;

node \*root;

tree()

{

root=NULL;

top=0;

}

void insert(int ch)

{

node \*temp,\*temp1;

if(root== NULL)

{

root=new node;

root->data=ch;

root->left=NULL;

root->right=NULL;

return;

}

temp1=new node;

temp1->data=ch;

temp1->right=temp1->left=NULL;

temp=search(root,ch);

if(temp->data>ch)

temp->left=temp1;

else

temp->right=temp1;

}

node \*search(node \*temp,int ch)

{

if(root== NULL)

{

cout <<"no node present";

return NULL;

}

if(temp->left==NULL && temp->right== NULL)

return temp;

if(temp->data>ch)

{ if(temp->left==NULL) return temp;

search(temp->left,ch);}

else

{ if(temp->right==NULL) return temp;

search(temp->right,ch);

} }

void display(node \*temp)

{

if(temp==NULL)

return ;

display(temp->left);

cout<<temp->data;

display(temp->right);

}

void inorder( node \*root)

{

node \*p;

p=root;

top=0;

do

{

while(p!=NULL)

{

stk[top]=p->data;

top++;

p=p->left;

}

if(top>0)

{

p=pop(root);

cout << p->data;

p=p->right;

}

}while(top!=0 || p!=NULL);

}

node \* pop(node \*p)

{

int ch;

ch=stk[top-1];

if(p->data==ch)

{

top--;

return p;

}

if(p->data>ch)

pop(p->left);

else

pop(p->right);

}

};

main()

{

tree t1;

int ch,n,i;

while(1)

{

cout <<"\n1.INSERT\n2.DISPLAY 3.INORDER TRAVERSE\n4.EXIT\nEnter your choice:";

cin >> ch;

switch(ch)

{

case 1: cout <<"enter no of elements to insert:";

cin >> n;

for(i=1;i<=n;i++)

{ cin >> ch;

t1.insert(ch);

}

break;

case 2: t1.display(t1.root);break;

case 3: t1.inorder(t1.root); break;

case 4: exit(1);

}

}

}

**OUTPUT**

1.INSERT  
2.DISPLAY 3.INORDER TRAVERSE  
4.EXIT  
Enter your choice:1  
enter no of elements to inser  
5 24 36 11 44 2 21

1.INSERT  
2.DISPLAY 3.INORDER TRAVERSE  
4.EXIT  
Enter your choice:3  
251121243644

1.INSERT  
2.DISPLAY 3.INORDER TRAVERSE  
4.EXIT  
Enter your choice:3  
251121243644

1.INSERT  
2.DISPLAY 3.INORDER TRAVERSE  
4.EXIT  
Enter your choice:4

**16. Write C++ programs for the depth first and breadth first traversals of a graph**.