**Q. Write a program in C++ to perform disk file sorting**.

#include<iostream>

#include<conio.h>

#include<fstream>

using namespace std;

const int SIZE=10;

int main()

{

int arr[SIZE];

int i=0,j=0;

int record;

ofstream out;

ifstream in;

out.open("Numbers.txt",ios::out);

cout<<"Enter 10 records\n";

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

{

cin>>record;

out<<record<<" ";

}

out.close();

in.open("Numbers.txt",ios::in);

cout<<"Values inserted in file are :"<<endl;

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

{

in>>record;

cout<<" "<<record<<" ";

arr[i]=record;

}

//Bubble sort used

int temp;

for(i=SIZE-1;i>=0;i--)

{

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

{

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

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

cout<<endl;

in.close();

out.open("Numbers.txt",ios::out);

cout<<"Values after sorting in the file are : "<<endl;

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

{

cout<<" "<<arr[i]<<" ";

record=arr[i];

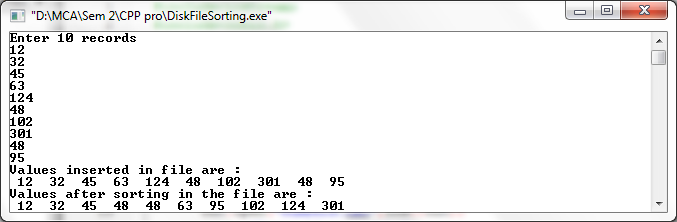
out<<record<<" ";

}

out.close();

};

**Output -**



**Q. Write a C++ program to perform evaluation of postfix expression.**

#include<iostream>

#include<conio.h>

#include<string.h>

#include<math.h>

using namespace std;

class postfix

{

public:

int top;

char p[50];

long int A[50];

postfix()

{

top=-1;

}

void input();

void push(long int s);

long int pop();

int full();

int empty();

long int eval\_post();

};

void postfix::input()

{

cout<<"enter a postfix expression\n";

cin>>p;

}

int postfix::full()

{

if(top==49)

return 1;

else

return 0;

}

void postfix::push(long int s)

{

if(full())

cout<<"overflow\n";

else

{

top=top+1;

A[top]=s;

}

}

int postfix::empty()

{

if(top==-1)

return 1;

else

return 0;

}

long int postfix::pop()

{

if(empty())

cout<<"underflow\n";

else

return(A[top--]);

}

long int postfix::eval\_post()

{

long int a,b,temp,result,len;

int i;

len=strlen(p);

p[len]='#';

for(i=0;p[i]!='#';i++)

{

if(p[i]<='9'&&p[i]>='0')

push(p[i]-48);

else

{

a=pop();

b=pop();

switch(p[i])

{

case '+':

temp=b+a;

break;

case '-':

temp=b-a;

break;

case '\*':

temp=b\*a;

break;

case '/':

temp=b/1;

break;

case '%':

temp=b%a;

break;

case '^':

temp=pow(b,a);

}

push(temp);

}

}

result=pop();

return result;

}

main()

{

long int value;

postfix f;

f.input();

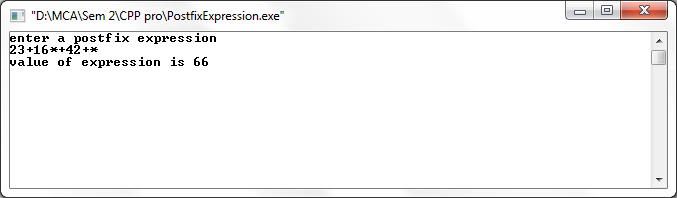
value=f.eval\_post();

cout<<"value of expression is "<<value;

getch();

}

**Output –**

**Q. Write a c++ program to perform following operations on two singly linked list.**

**1.Merge**

**2.Append**

**3.Union**

**4.Intersection**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

Node \*next;

int data;

}

\*head1=NULL,\*head2=NULL;

void disp(Node \*h)

{

Node \*tmp;

tmp=h;

while(tmp!=NULL)

{

cout<<tmp->data<<" | ";

tmp=tmp->next;

}

cout<<"\n";

}

void add()

{

int times;

cout<<"\nEnter the number of elements in First List"<<"\n";

cin>>times;

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

{

cout<<"Enter data"<<"\n";

int dat;

cin>>dat;

Node \*tmp,\*q,\*p;

tmp=new Node;

q=head1;

tmp->data=dat;

tmp->next=NULL;

if(head1==NULL)

{

head1=tmp;

}

else

{

while(q!=NULL)

{

if(q->next==NULL)

{

p=q;

}

q=q->next;

}

p->next=tmp;

}

}

cout<<"Elements in List one are : "<<"\n";

disp(head1);

cout<<"\nEnter the number of elements in Second List"<<"\n";

cin>>times;

for(int j=0;j<times;j++)

{

cout<<"Enter data"<<"\n";

int dat;

cin>>dat;

Node \*tmp,\*q,\*p;

tmp=new Node;

q=head2;

tmp->data=dat;

tmp->next=NULL;

if(head2==NULL)

{

head2=tmp;

}

else

{

while(q!=NULL)

{

if(q->next==NULL)

{

p=q;

}

q=q->next;

}

p->next=tmp;

}

}

cout<<"Elements in List two are : "<<"\n";

disp(head2);

}

void intersection()

{

Node \*tmp1,\*tmp2,\*tmp,\*ptr,\*ptr1,\*tn;

tmp1=head1;

tmp2=head2;

tmp=NULL;

tn=NULL;

ptr1=NULL;

while(tmp1!=NULL)

{

tn=tmp2;

while(tn!=NULL)

{

if(tmp1->data==tn->data)

{

tmp=new Node;

tmp->next=NULL;

tmp->data=tmp1->data;

}

tn=tn->next;

}

tmp1=tmp1->next;

}

cout<<"\nLinked List after insertion is : "<<"\n";

disp(tmp);

}

void merge()

{

Node \*tmp1,\*tmp2,\*p,\*q;

tmp1=head1;

tmp2=head2;

head2=NULL;

delete head2;

while(tmp1!=NULL && tmp2!=NULL)

{

p=tmp1->next;

tmp1->next=tmp2;

q=tmp2->next;

if(p!=NULL)

tmp2->next=p;

tmp1=p;

tmp2=q;

}

cout<<"\nLinked List after Merge is : "<<"\n";

disp(head1);

}

void unions()

{

Node \*tmp1,\*tmp2,\*tmp,\*q,\*p;

int c=0;

tmp1=head1;

tmp2=head2;

p=NULL;

q=NULL;

tmp=NULL;

while(tmp1!=NULL)

{

tmp=new Node;

tmp->next=NULL;

tmp->data=tmp1->data;

tmp1=tmp1->next;

if(p==NULL)

p=tmp;

else

q->next=tmp;

q=tmp;

}

tmp1=head1;

while(tmp2!=NULL)

{

while(tmp1!=NULL)

{

if(tmp1->data==tmp2->data)

goto a;

tmp1=tmp1->next;

}

tmp=new Node;

tmp->next=NULL;

tmp->data=tmp2->data;

if(p==NULL)

p=tmp;

else

q->next=tmp;

q=tmp;

a:

tmp2=tmp2->next;

}

q->next=NULL;

cout<<"\nLinked List after Union is : "<<"\n";

disp(p);

}

void append()

{

Node\* tmp1,\*p;

tmp1=head1;

while(tmp1!=NULL)

{

p=tmp1;

tmp1=tmp1->next;

if(tmp1==NULL)

{

p->next=head2;

}

}

cout<<"\nLinked List after Append is : "<<"\n";

disp(head1);

delete head2;

}

int main()

{

int ch;

while(ch!=7)

{

cout<<"\n1.Add\n2.Merge\n3.Union\n4.Insertion\n5.Append\n";

cout<<"\nEnter choice"<<"\n";

cin>>ch;

switch(ch)

{

case 1:

add();

break;

case 2:

merge();

break;

case 3:

unions();

break;

case 4:

intersection();

break;

case 5:

append();

break;

case 6:

exit(0);

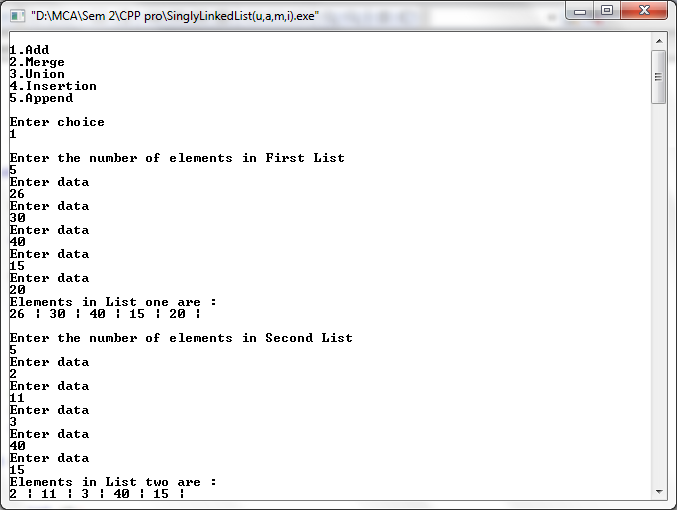
}

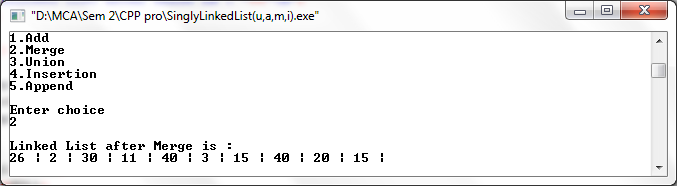
}

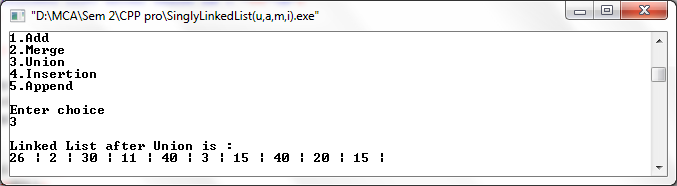
return 0;

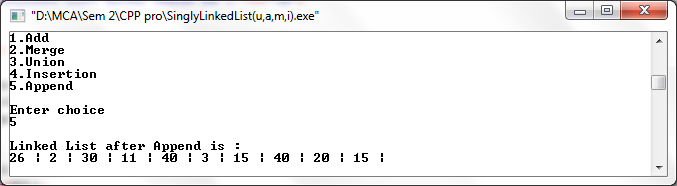
}

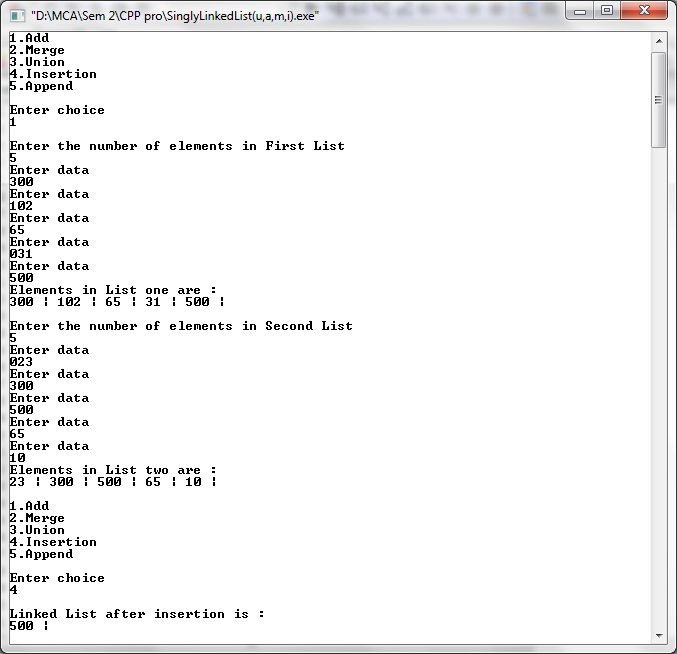
**Output –**



****

****

****

****

**Q. Write a c++ program to perform heap sort.**

#include <iostream>

#include <conio.h>

#include <cstdlib>

#include <time.h>

using namespace std;

const int LOW = 1;

const int HIGH = 100;

void max\_heapify(int \*a, int i, int n)

{

int j, temp;

temp = a[i];

j = 2 \* i;

while (j <= n)

{

if (j < n && a[j + 1] > a[j])

j = j + 1;

if (temp > a[j])

break;

else if (temp <= a[j])

{

a[j / 2] = a[j];

j = 2 \* j;

}

}

a[j / 2] = temp;

return;

}

void heapsort(int \*a, int n)

{

int i, temp;

for (i = n; i >= 2; i--)

{

temp = a[i];

a[i] = a[1];

a[1] = temp;

max\_heapify(a, 1, i - 1);

}

}

void build\_maxheap(int \*a, int n)

{

int i;

for (i = n / 2; i >= 1; i--)

{

max\_heapify(a, i, n);

}

}

int main()

{

int n, i;

cout << "Enter no of elements to be sorted:";

cin >> n;

int a[n];

cout << "Elements are:\n";

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

{

cin >> a[i] ;

}

build\_maxheap(a, n);

heapsort(a, n);

cout << "\nSorted elements are:\n";

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

{

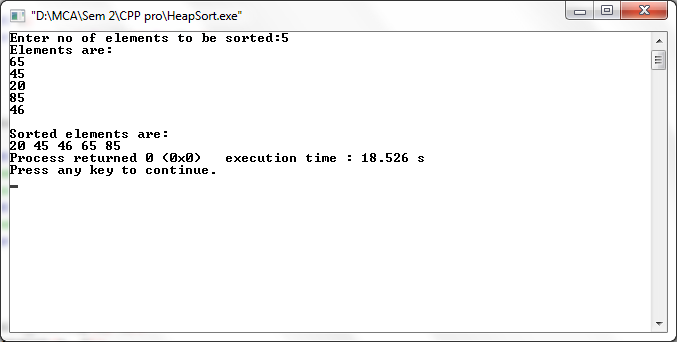
cout << a[i] << " ";

}

return 0;

}

**Output –**

****

**Q. Write a menu driven c++ program to perform following hashing techniques and use linear**

**probing for hash collision resolution**

**- Direct hashing**

**- Subtraction hashing**

**- Modulo division**

**- Folding (shift,boundary)**

**- Digit extraction.**

#include<iostream>

#include<conio.h>

#include<stdio.h>

#include<iomanip>

using namespace std;

const int SIZE=10;

static int coll;

class hash1

{

long key;

long index;

long arr[10];

public:

void directHash();

void subHash();

void modDivision();

void linProbe();

void digitExHash();

void foldShiftHash();

void foldBoundHash();

void display();

};

void hash1::modDivision()

{

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

arr[i]=-1;

for(int i=1;i<=7;i++)

{

int x;

cout<<"\nEnter a number";

cin>>x;

index=x%10;

while(arr[index]!=-1)

index=(index+1)%10;

arr[index]=x;

}

}

void hash1::display()

{

cout<<"\nHASH TABLE\n";

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

cout<<setw(8)<<i;

cout<<"\n";

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

cout<<setw(8)<<arr [i];

}

void hash1::directHash()

{

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

arr[i]=-1;

for(int i=1;i<=10;i++)

{

int x;

cout<<"Enter numbers from 1 to 10\n";

cin>>x;

int index=x;

arr[index]=x;

}

}

void hash1::subHash()

{

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

arr[i]=-1;

for(int i=1;i<=7;i++)

{

int x;

cout<<"Enter numbers from 1001 to 1010\n";

cin>>x;

int index=x-1000;

arr[index]=x;

}

}

void hash1::digitExHash()

{

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

arr[i]=-1;

for(int i=1;i<=10;i++)

{

int x;

cout<<"Enter a number of 6 digits\n";

cin>>x;

int index=0;

long r,inc=100000,incr=1000;

for(int i=1;i<=6;i++)

{

if(i==1 || i== 3 || i==5)

{

incr=incr/10;

r=(x/inc)%10;

index=index+(r\*incr);

}

inc=inc/10;

}

index=index%10;

while(arr[index]!=-1)

index=(index+1)%10;

arr[index]=x;

}

}

void hash1::foldShiftHash()

{

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

arr[i]=-1;

for(int i=1;i<=10;i++)

{

int x;

cout<<"Enter a number of 4 digits\n";

cin>>x;

index=0;

long no,no1,no2,no3;

no1=x/100;

no3=no1\*100;

no2=x%no3;

index=no1+no2;

index=index%10;

if(index==-1)

{

arr[index]=x;

}

while(arr[index]!=-1)

index=(index+1)%10;

arr[index]=x;

}

}

void hash1::foldBoundHash()

{

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

arr[i]=-1;

for(int i=1;i<=10;i++)

{

int x;

cout<<"Enter a number of 4 digits\n";

cin>>x;

index=0;

long no,no1,no2,no3;

no1=x/100;

no3=no1\*100;

no2=x%no3;

int tmp=0;

while(no1>0)

{

int rem=no1%10;

tmp=(tmp\*10)+rem;

no1=no1/10;

}

int tmp1=0;

while(no2>0)

{

int rem1=no2%10;

tmp1=(tmp1\*10)+rem1;

no2=no2/10;

}

index=tmp+tmp1;

index=index%10;

if(index==-1)

{

arr[index]=x;

}

while(arr[index]!=-1)

index=(index+1)%10;

arr[index]=x;

}

}

int main()

{

hash1 h;

int op;

cout<<"Enter 1 for direct hashing\nEnter 2 for Subtraction Hashing\nEnter 3 for Modulo Division Hashing"<<endl;

cout<<"Enter 4 for digit extraction hashing\nEnter 5 for shift fold Hashing\nEnter 6 for shift Boundry Hashing"<<endl;

cout<<"\nEnter 7 to exit\n"<<endl;

cin>>op;

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

{

switch(op)

{

case 1:

h.directHash();

h.display();

break;

case 2:

h.subHash();

h.display();

break;

case 3:

h.modDivision();

h.display();

break;

case 4:

h.digitExHash();

h.display();

break;

case 5:

h.foldShiftHash();

h.display();

break;

case 6:

h.foldBoundHash();

h.display();

break;

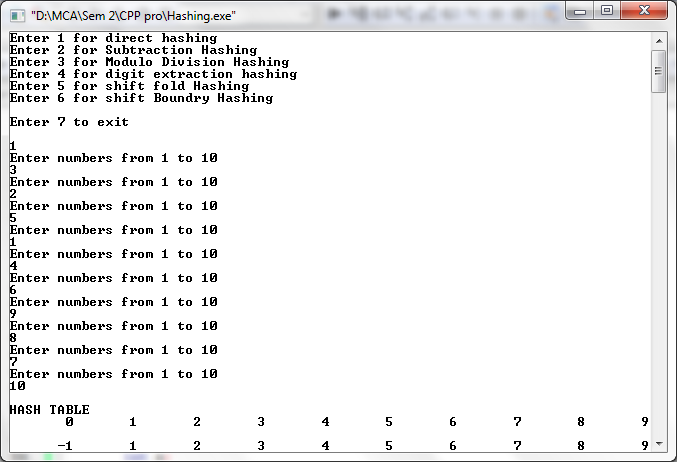
}

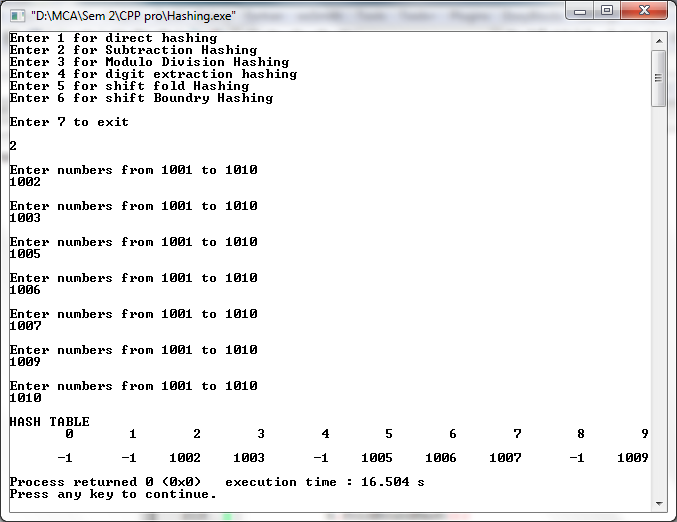
}

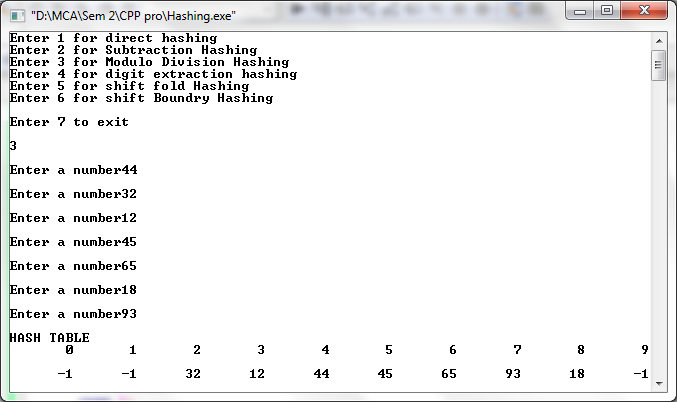
return 0;

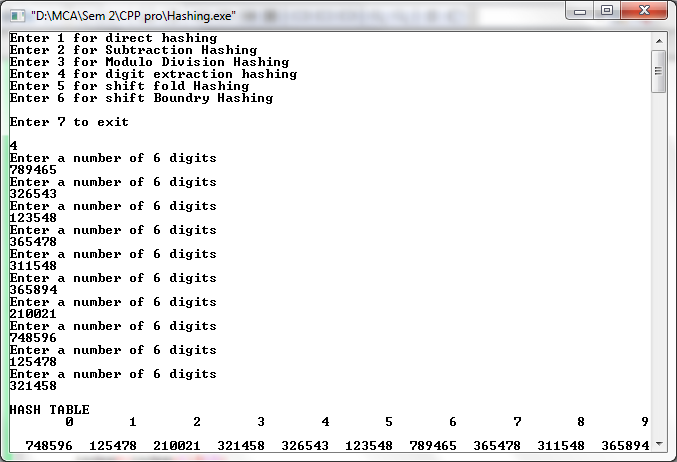
}

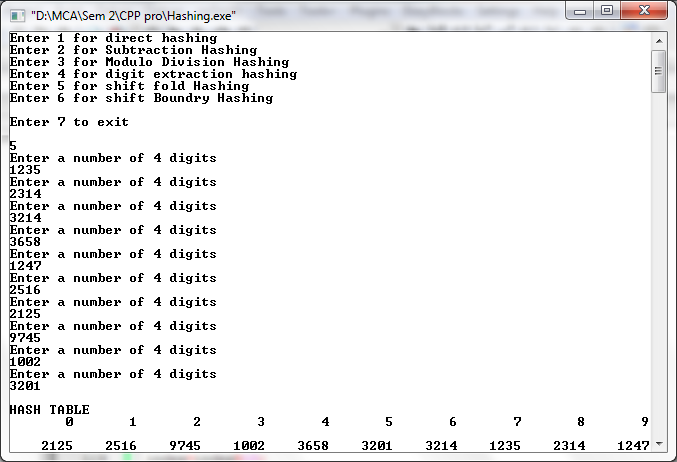
**Output –**

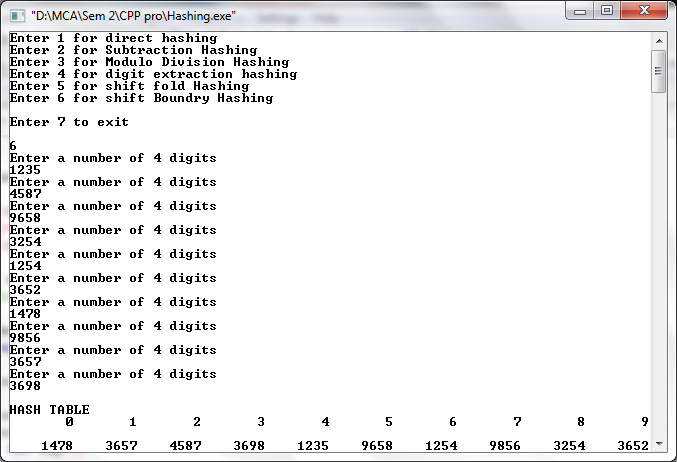


****









**Q. Write a C++ program to implement Depth First Traversal and Breadth First Traversal.**

#include<iostream>

#include<stdio.h>

#define max 20

using namespace std;

int adj[max][max];

bool visited[max];

int n;

int frnt;

void create\_graph()

{

int i, max\_edges,origin,destin;

cout<<"Enter no. of nodes: ";

cin>>n;

max\_edges=n\*(n-1);

for(i=1;i<=max\_edges;i++)

{

cout<<"Enter edge (0 0 to quit) : "<<i<<"\n";

cin>>origin>>destin;

if(origin==0||destin==0)

break;

if(origin>n||destin>n||origin<=0||destin<=0)

{

cout<<"Invalid edge \n";

i--;

}

else

{

adj[origin][destin]=1;

}

}

}

void display()

{

int i, j;

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

{

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

{

cout<<adj[i][j]<<"\t";

}

cout<<"\n";

}

}

void dfs(int v)

{

int i, stack[max], top=-1,pop\_v, j, t;

int c;

top++;

stack[top]=v;

while(top>=0)

{

pop\_v=stack[top];

top--;

if(visited[pop\_v]==false)

{

cout<<pop\_v;

visited[pop\_v]=true;

}

else

continue;

for(i=n;i>=1;i--)

{

if(adj[pop\_v][i]==1 && visited[i]==false)

{

top++;

stack[top]=i;

}

}

}

}

void bfs(int v)

{

int i, frnt, rear;

int que[20];

frnt=rear=-1;

cout<<v;

visited[v]=true;

rear++;

frnt++;

que[rear]=v;

while(frnt<=rear)

{

v=que[frnt];

frnt++;

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

{

if(adj[v][i]==1&&visited[i]==false)

{

cout<<i<<"\t";

visited[i]=true;

rear++;

que[rear]=i;

}

}

}

}

void adj\_nodes(int v)

{

int i;

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

{

int i;

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

{

if(adj[v][i]==1)

cout<<i;

cout<<"\n";

}

}

}

int main()

{

int i, v, ch;

create\_graph();

while(1)

{

cout<<"\n";

cout<<"1. Adjacency Matrix \n";

cout<<"2. Depth first search using stack\n";

cout<<"3. Breadth first search\n";

cout<<"4. exit \n";

cout<<"Enter your choice\n";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Adjacency Matrix \n";

display();

break;

case 2:

cout<<"Enter starting node for Depth First Search: \n";

cin>>v;

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

visited[i]=false;

dfs(v);

break;

case 3:

cout<<"Enter starting node for Breadth First Search: \n";

cin>>v;

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

visited[i]=false;

bfs(v);

break;

case 4:

break;

default:

cout<<"Wrong Choice";

break;

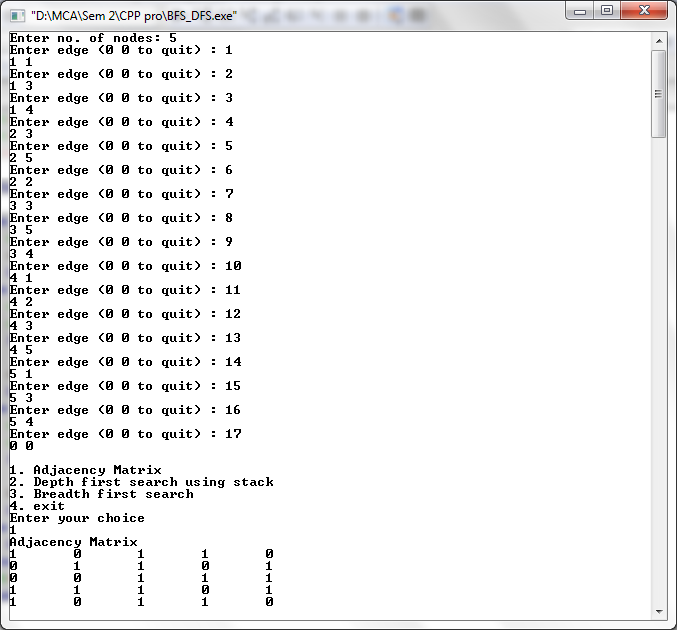
}

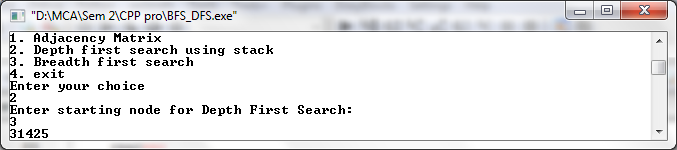
}

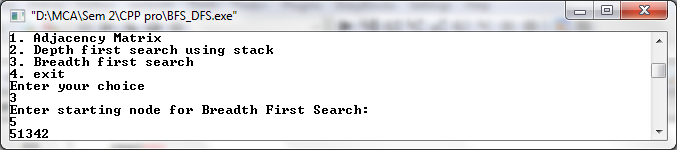
return 0;

}

**Output –**

****

****

****