**PROGRAM NO.1**

**/\***

**PROGRAM NAME= WRITE A PROGRAM TO SORT POSITIVE INTEGER NUMBERS IN NON-DECREASING ORDER USING INSERTION SORT.**

**ROLL NO. = 17103011**

**DATE = 09/01/2019 \*/**

#include<iostream>

using namespace std;

int main()

{

int testcase;

cout<<"enter no. of testcases\n";

cin>>testcase;

while(testcase--)

{

int i,j,flag=0,n,key,loop\_counter=0;

cout<<"enter size of array\n";

cin>>n;

int a[n];

cout<<"enter array\n";

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

{

cin>>a[i];

if(a[i]<0)

flag=1;

}

if(flag)

cout<<"invalid input\n";

else

{

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

{

loop\_counter++;

key=a[i+1];

for(j=i;j>=0;j--)

{

loop\_counter++;

if(key<a[j])

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

else

break;

}

a[j+1]=key;

}

cout<<"sorted array is \n";

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

{

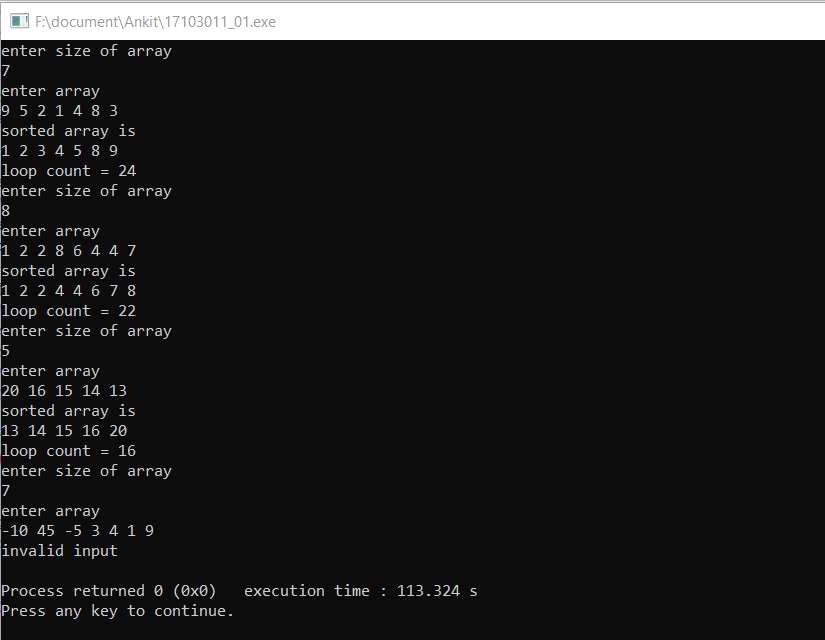
cout<<a[i]<<" ";

}

cout<<"\nloop count = "<<loop\_counter<<"\n";}

}

}



**PROGRAM NO.2**

**/\***

**PROGRAM NAME= WRITE A PROGRAM TO SORT POSITIVE INTEGER NUMBERS IN NON-DECREASING ORDER USING SELECTION SORT.**

**ROLL NO. = 17103011**

**DATE = 09/01/2019 \*/**

#include<iostream>

using namespace std;

int main()

{

int testcase;

cout<<"enter no. of testcases\n";

cin>>testcase;

while(testcase--)

{

int i,j,flag=0,k,n,minimum,loop\_counter=0;

cout<<"enter size of array\n";

cin>>n;

int a[n];

cout<<"enter array\n";

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

{

cin>>a[i];

if(a[i]<0)

flag=1;

}

if(flag)

cout<<"invalid input\n";

else

{

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

{

loop\_counter++;

k=i;

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

{

loop\_counter++;

if(a[j]<a[k])

{

k=j;

}

}

j=a[k];

a[k]=a[i];

a[i]=j;

}

cout<<"sorted array is \n";

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

{

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

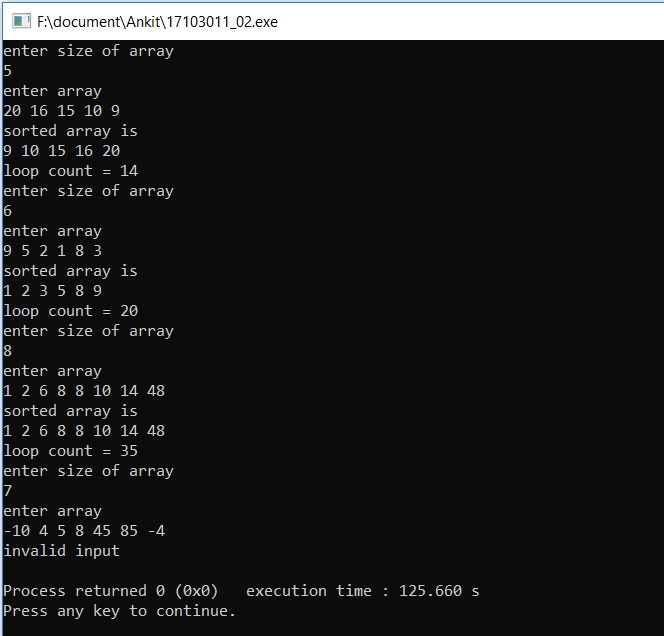
}

cout<<"\nloop count = "<<loop\_counter<<"\n";

}

}

}



**PROGRAM NO.3**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO SORT INTEGER NUMBERS IN NON-DECREASING ORDER USING MERGE SORT**

**ROLL NO. = 17103011**

**DATE = 16/01/2019 \*/**

#include<iostream>

using namespace std;

int counter;

void merge(int arr[],int lower,int mid,int uper)

{

int size=uper-lower+1;

int size\_left=mid-lower+1;

int size\_right=uper-mid;

int array\_left[size\_left+1],array\_right[size\_right+1];

int i,j,k;

for(i=0;i<size\_left;i++)

array\_left[i]=arr[lower+i];

for(j=0;j<size\_right;j++)

array\_right[j]=arr[mid+1+j];

array\_left[i]=INT\_MAX;

array\_right[j]=INT\_MAX;

i=j=0;

for(k=0;k<size;k++)

{

counter++;

if(array\_left[i]<=array\_right[j])

{

arr[lower+k]=array\_left[i];

i++;

}

else

{

arr[lower+k]=array\_right[j];

j++;

}

}

}

void merge\_sort(int arr[],int l\_bound,int u\_bound)

{

if(l\_bound<u\_bound)

{

int mid=(l\_bound+u\_bound)/2;

merge\_sort(arr,l\_bound,mid);

merge\_sort(arr,mid+1,u\_bound);

merge(arr,l\_bound,mid,u\_bound);

}

}

int main()

{

int testcases;

cout<<"enter no. of testcases\n";

cin>>testcases;

while(testcases--)

{

int size,i=0,j,flag=0;

cout<<"enter size of array\n";

cin>>size;

int arr[size];

cout<<"enter array\n";

cin>>arr[i];

i++;

if(arr[0]>=0)

{

flag=1;

while(i<size)

{

cin>>arr[i];

if(arr[i]<0)

flag=3;

i++;

}

}

else

{

flag=2;

while(i<size)

{

cin>>arr[i];

if(arr[i]>=0)

flag=3;

i++;

}

}

if(flag==3)

cout<<"invalid input\n";

else

{

counter=0;

merge\_sort(arr,0,size-1);

i=0;

cout<<"sorted array is \n";

while(i<size)

{

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

i++;

}

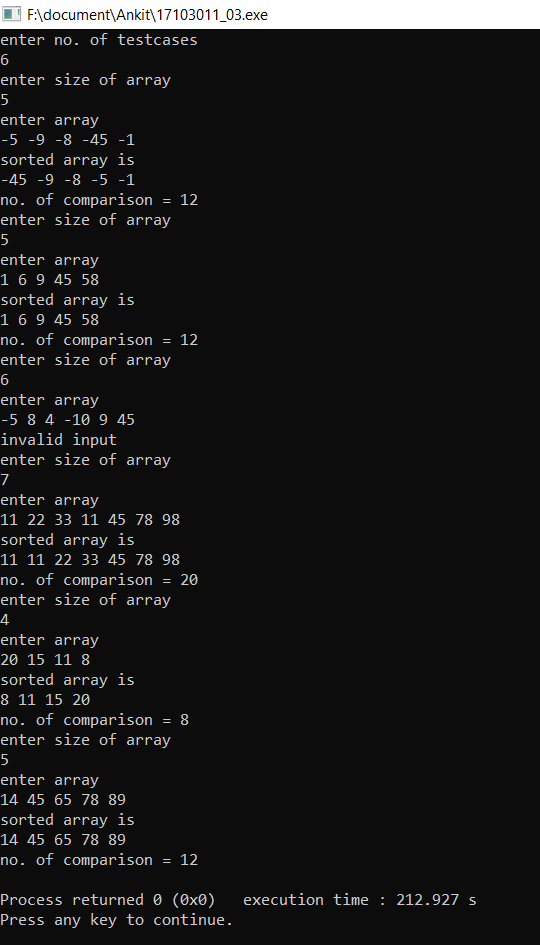
cout<<"\nno. of comparison = "<<counter;

cout<<"\n";

}

}

}



**PROGRAM NO.4**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO SORT NON REPEATING INTEGERS IN NON DECREASING ORDER USING QUICK SORT**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 23/01/2019 \*/**

#include<bits/stdc++.h>

#include<map>

using namespace std;

int partition(int[],int,int);

void quick\_sort(int array[],int first,int last)

{

if(first<last)

{

int pivot=partition(array,first,last);

quick\_sort(array,first,pivot-1);

quick\_sort(array,pivot+1,last);

}

}

int comparision;

int partition(int array[],int low,int high)

{

int i,j,k;

i=low-1;

int pivot=array[high];

for(j=low;j<high;j++)

{

comparision++;

if(array[j]<pivot)

{

i++;

k=array[j];

array[j]=array[i];

array[i]=k;

}

}

k=array[i+1];

array[i+1]=array[high];

array[high]=k;

return (i+1);

}

int main()

{

int testcases;

cout<<"enter number of testcases\n";

cin>>testcases;

while(testcases--)

{

comparision=0;

int i,j,size,flag=0;

map<int,int> m;

cout<<"enter size of array\n";

cin>>size;

int array[size];

i=0;

cout<<"enter array\n";

while(i<size)

{

cin>>array[i];

m[array[i]]++;

if(m[array[i]]>1)

{

flag=1;

}

i++;

}

if(flag==1)

cout<<"invalid input\n";

else

{

quick\_sort(array,0,size-1);

cout<<"sorted array is\n";

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

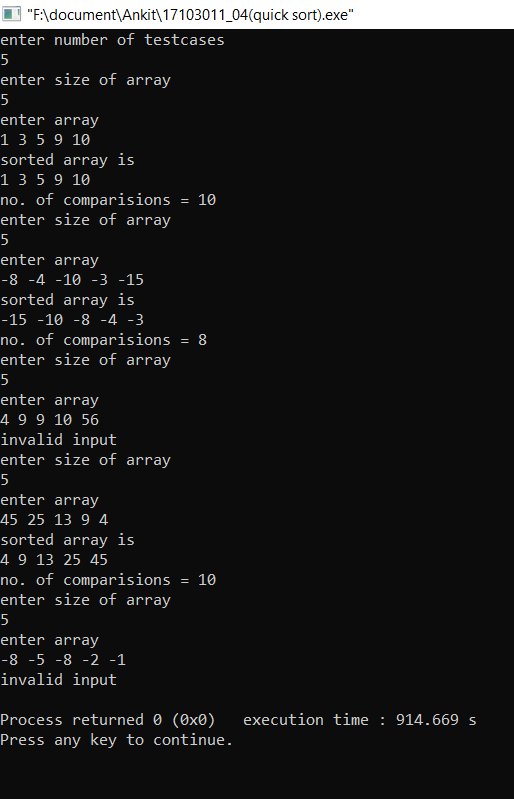
cout<<array[i]<<" ";

cout<<"\nno. of comparisions = "<<comparision<<"\n";

}

}

}



**PROGRAM NO.5**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO SORT INTEGERS IN NON DECREASING ORDER USING HEAP SORT**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 30/01/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

void max\_heapify(int array[],int i,int size)

{

int j,left,right,largest;

left=2\*i+1;

right=2\*i+2;

largest=i;

if(array[largest]<array[left]&&left<size)

{

largest=left;

}

if(array[largest]<array[right]&&right<size)

{

largest=right;

}

if(largest!=i)

{

j=array[largest];

array[largest]=array[i];

array[i]=j;

max\_heapify(array,largest,size);

}

}

int loop\_count;

void heap\_sort(int array[],int size)

{

int i,j;

for(i=size/2-1;i>=0;i--)

{

loop\_count++;

max\_heapify(array,i,size);

}

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

{

loop\_count++;

j=array[0];

array[0]=array[i];

array[i]=j;

max\_heapify(array,0,i);

}

}

int main()

{

int testcases;

cout<<"enter no. of testcases\n";

cin>>testcases;

while(testcases--)

{

int i,j,k,size;

loop\_count=0;

cout<<"enter no. of elements\n";

cin>>size;

int array[size];

i=0;

cout<<"enter array\n";

while(i<size)

{

cin>>array[i];

i++;

}

heap\_sort(array,size);

cout<<"sorted array is \n";

i=0;

while(i<size)

{

cout<<array[i]<<" ";

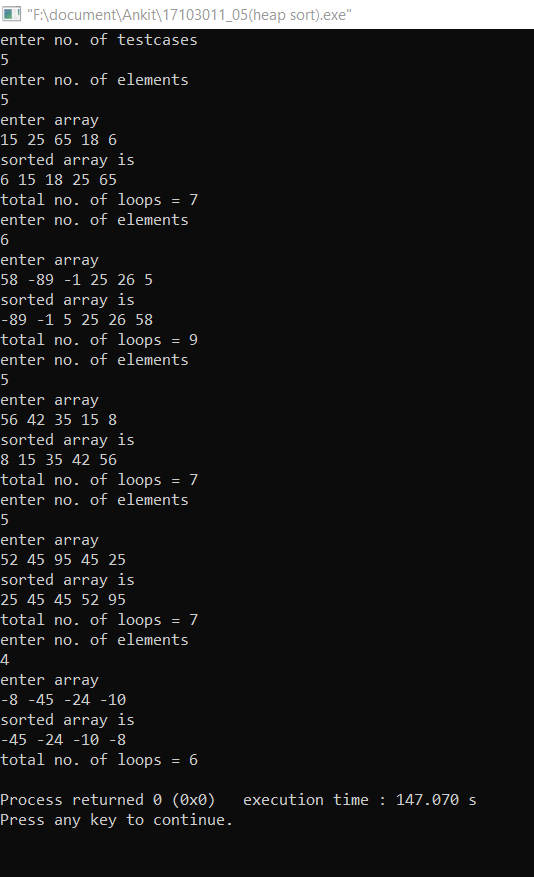
i++;

}

cout<<"\ntotal no. of loops = "<<loop\_count<<"\n";

}

}



**PROGRAM NO.6**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT PRIORITY QUEUE USING MIN HEAP**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 13/02/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

void min\_heapify(int array[],int i,int size)

{

int j,left,right,least;

left=2\*i+1;

right=2\*i+2;

least=i;

if(array[least]>array[left]&&left<size)

least=left;

if(array[least]>array[right]&&right<size)

least=right;

if(least!=i)

{

j=array[least];

array[least]=array[i];

array[i]=j;

min\_heapify(array,least,size);

}

}

int minimum(int array[])

{

return array[0];

}

int extract\_min(int array[],int &size)

{

int minimum=array[0];

array[0]=array[size-1];

size=size-1;

min\_heapify(array,0,size);

return minimum;

}

void decrease\_key(int array[],int index,int key)

{

if(key>array[index])

cout<<"new key is larger than previous key\n";

else

{

array[index]=key;

while(index>0&&key<array[(index-1)/2])

{

int j=array[index];

array[index]=array[(index-1)/2];

array[(index-1)/2]=j;

index=(index-1)/2;

}

}

}

void insert(int array[],int key,int &size)

{

size=size+1;

array[size-1]=INT\_MAX;

decrease\_key(array,size-1,key);

}

void build\_minheap(int array[],int size)

{

int i;

for(i=size/2-1;i>=0;i--)

min\_heapify(array,i,size);

}

int main()

{

int i=0,temp,size;

cout<<"enter no. of elements\n";

cin>>size;

int array[size];

cout<<"enter array\n";

while(i<size)

{

cin>>array[i];

i++;

}

build\_minheap(array,size);

i=0;

while(i<size)

{

cout<<array[i]<<" ";

i++;

}

cout<<"\npress 1 for return minimum value\npress 2 for extract minimum element\n";

cout<<"press 3 for decrease key\npress 4 for insert a new key\npress 6 to print queue\n";

cout<<"press 5 for exit\nenter choice\n";

int choice,index,key;

cin>>choice;

while(choice!=5)

{

switch(choice)

{

case 1:

temp=minimum(array);

cout<<"minimum value is "<<temp<<"\n";

break;

case 2:

temp=extract\_min(array,size);

cout<<"minimum extracted value is "<<temp<<"\n";

break;

case 3:

cout<<"enter index and key \n";

cin>>index>>key;

decrease\_key(array,index-1,key);

break;

case 4:

cout<<"enter new key to insert\n";

cin>>key;

insert(array,key,size);

break;

case 6:

i=0;

while(i<size)

{

cout<<array[i]<<" ";

i++;

}

cout<<"\n";

break;

default:

cout<<"invalid input\n";

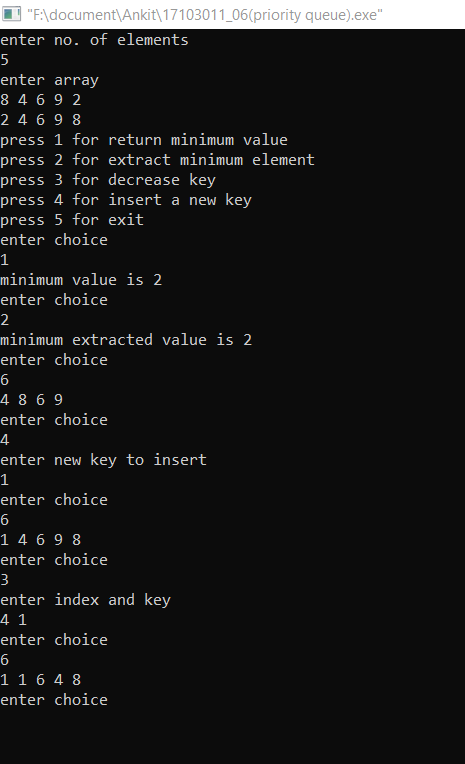
}

cout<<"enter choice\n";

cin>>choice;

}

}



**PROGRAM NO.7**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO SORT IN NON DECREASING ORDER USING COUNTING SORT**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 20/02/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int testcases;

cout<<"enter no. of testcases\n";

cin>>testcases;

while(testcases--)

{

int size,i,max,min,range;

cout<<"enter size of array\n";

cin>>size;

int array[size];

cout<<"enter array\n";

cin>>array[0];

min=max=array[0];

i=1;

while(i<size)

{

cin>>array[i];

if(array[i]>max)

max=array[i];

if(array[i]<min)

min=array[i];

i++;

}

range=max-min+1;

int arr\_range[range]={0};

i=0;

while(i<size)

{

arr\_range[array[i]-min]+=1;

i++;

}

i=0;

int element=min-1,freq=0;

while(i<size)

{

if(freq==0)

{

element++;

freq=arr\_range[element-min];

}

else

{

array[i]=element;

freq--;

i++;

}

}

cout<<"sorted array is \n";

i=0;

while(i<size)

{

cout<<array[i]<<" ";

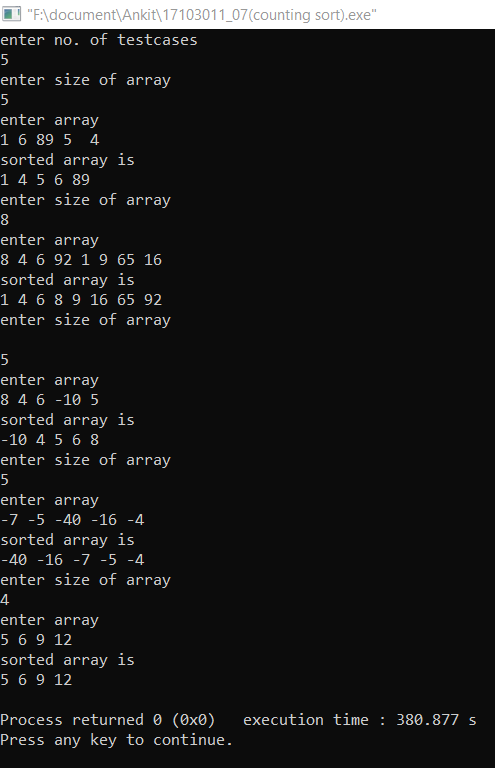
i++;

}

cout<<"\n";

}

}



**PROGRAM NO.8**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO SORT IN NON DECREASING ORDER USING RADIX SORT**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 20/02/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

void counting\_sort(int array[],int factor,int size)

{

int i=0,count\_array[10]={0};

while(i<size)

{

count\_array[(array[i]/factor)%10]++;

i++;

}

i=1;

while(i<10)

{

count\_array[i]+=count\_array[i-1];

i++;

}

i=size-1;

int temp\_array[size];

while(i>=0)

{

temp\_array[count\_array[(array[i]/factor)%10]-1]=array[i];

count\_array[(array[i]/factor)%10]--;

i--;

}

i=0;

while(i<size)

{

array[i]=temp\_array[i];

i++;

}

}

int main()

{

int testcases;

cout<<"enter no. of testcases\n";

cin>>testcases;

while(testcases--)

{

int size,i,max;

cout<<"enter size of array\n";

cin>>size;

int array[size];

cout<<"enter array\n";

cin>>array[0];

max=array[0];

i=1;

while(i<size)

{

cin>>array[i];

if(array[i]>max)

max=array[i];

i++;

}

int max\_length=0;

while(max>0)

{

max=max/10;

max\_length++;

}

int factor=1;

while(max\_length--)

{

counting\_sort(array,factor,size);

factor=factor\*10;

}

cout<<"sorted array is \n";

i=0;

while(i<size)

{

cout<<array[i]<<" ";

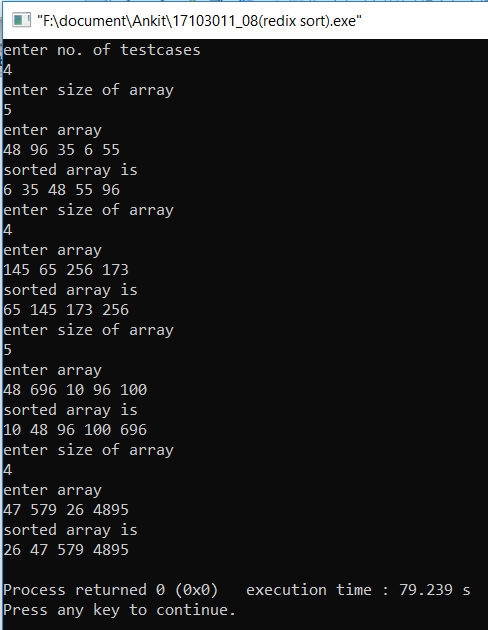
i++;

}

cout<<"\n";

}

}



**PROGRAM NO.9**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT HASH FUNCTION (DIVISION METHOD)**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 27/02/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

struct node

{

int info;

node \*next;

};

int main()

{

int size,i,choice,temp1,temp2 ,flag=0,count=0;;

node \*p1,\*p2;

cout<<"press 1 for insert element\npress 2 for searching element\n";

cout<<"press 3 for exit\nenter choice\n";

cin>>choice;

node \*array[100]={NULL};

while(choice!=3)

{

switch(choice)

{

case 1:

cout<<"enter number\n";

cin>>temp1;

p1=new node;

p1->info=temp1;

p1->next=NULL;

temp1=temp1%100;

if(array[temp1]==NULL)

array[temp1]=p1;

else

{

p2=array[temp1];

while(p2->next!=NULL)

p2=p2->next;

p2->next=p1;

}

break;

case 2:

cout<<"enter number\n";

cin>>temp1;

temp2=temp1%100;

p1=array[temp2];

flag=0,count=0;

while(p1!=NULL)

{

count++;

if(p1->info==temp1)

{

flag=1;

break;

}

p1=p1->next;

}

if(count>1)

cout<<"collision occured\n";

if(flag==1)

cout<<temp1<<" is present in the table\n";

else

cout<<temp1<<" is not present in the table\n";

break;

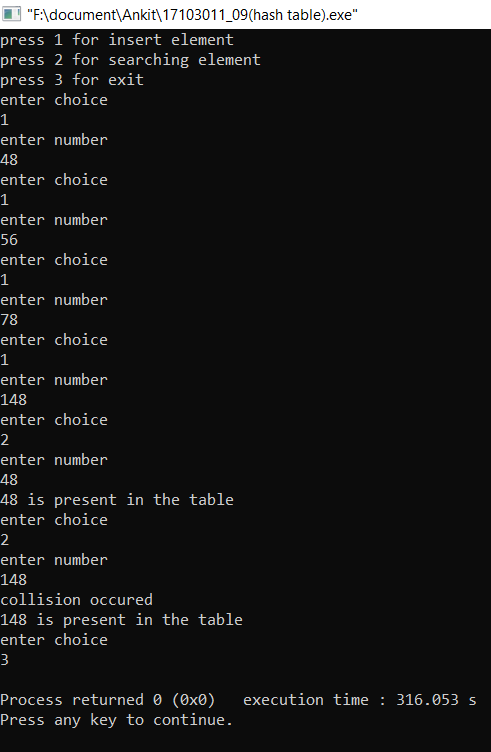
}

cout<<"enter choice\n";

cin>>choice;

}

}



**PROGRAM NO.10**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT HASH FUNCTION (MULTIPLICATION METHOD)**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 27/02/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

struct node

{

int info;

node \*next;

};

int main()

{

int size,i,choice,temp1,temp2,hash;

float temp;

node \*p1,\*p2;

cout<<"press 1 for insert element\npress 2 for searching element\n";

cout<<"press 3 for exit\nenter choice\n";

cin>>choice;

node \*array[100]={NULL};

int flag=0,count=0;

while(choice!=3)

{

switch(choice)

{

case 1:

cout<<"enter number\n";

cin>>temp1;

p1=new node;

p1->info=temp1;

p1->next=NULL;

temp=temp1\*0.43;

temp2=temp;

hash=64\*(temp-temp2);

temp1=hash;

if(array[temp1]==NULL)

array[temp1]=p1;

else

{

p2=array[temp1];

while(p2->next!=NULL)

p2=p2->next;

p2->next=p1;

}

break;

case 2:

cout<<"enter number\n";

cin>>temp1;

temp=temp1\*0.43;

temp2=temp;

hash=64\*(temp-temp2);

temp2=hash;

flag=0,count=0;

p1=array[temp2];

while(p1!=NULL)

{

count++;

if(p1->info==temp1)

{

flag=1;

break;

}

p1=p1->next;

}

if(count>1)

cout<<"collision occured\n";

if(flag==1)

cout<<temp1<<" is present in the table\n";

else

cout<<temp1<<" is not present in the table\n";

break;

default:

cout<<"invalid choice\n";

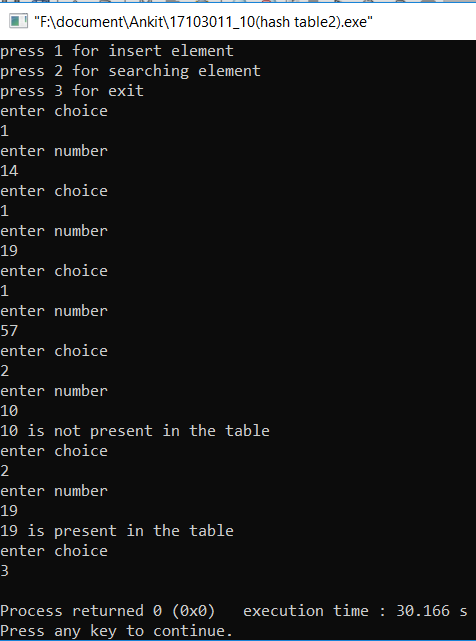
}

cout<<"enter choice\n";

cin>>choice;

}

}



**PROGRAM NO.11**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT INSERTION IN RED BLACK TREE**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 06/03/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

struct node

{

int info;

char colour;

node \*left,\*right,\*parent;

};

node \*root='\0';

void insert\_node(node \*&root,node \*p1)

{

if(root=='\0')

root=p1;

else

{

if(p1->info<root->info)

{

if(root->left=='\0')

{

p1->parent=root;

root->left=p1;

}

else

insert\_node(root->left,p1);

}

else

{

if(root->right=='\0')

{

p1->parent=root;

root->right=p1;

}

else

insert\_node(root->right,p1);

}

}

}

void rotate\_left(node \*t)

{

node \*child=t->left;

t->left=child->right;

if(t->left!='\0')

t->left->parent=t;

child->right=t;

node \*temp=t->parent;

t->parent=child;

if(temp=='\0')

{

child->parent='\0';

root=child;

}

else if(temp->left==t)

{

temp->left=child;

child->parent=temp;

}

else

{

temp->right=child;

child->parent=temp;

}

}

void rotate\_right(node \*t)

{

node \*child=t->right;

t->right=child->left;

if(t->right!='\0')

t->right->parent=t;

child->left=t;

node \*temp=t->parent;

t->parent=child;

if(temp=='\0')

{

child->parent='\0';

root=child;

}

else if(temp->left==t)

{

temp->left=child;

child->parent=temp;

}

else

{

temp->right=child;

child->parent=temp;

}

}

void set\_node(node \*&root,node \*&t)

{

if(t->parent=='\0')

{

t->colour='b';

return;

}

while(t->parent!='\0'&&t->parent->colour!='b')

{

node \*uncle,\*par=t->parent,\*g\_par=t->parent->parent;

if(g\_par->left==par)

uncle=g\_par->right;

else

uncle=g\_par->left;

if(uncle!='\0'&&uncle->colour=='r')

{

par->colour='b';

uncle->colour='b';

g\_par->colour='r';

t=g\_par;

}

else

{

if(g\_par->left==par)

{

if(par->right==t)

{

t=t->parent;

rotate\_right(t);

}

t->parent->colour='b';

g\_par->colour='r';

rotate\_left(g\_par);

}

else

{

if(par->left==t)

{

t=t->parent;

rotate\_left(t);

}

t->parent->colour='b';

g\_par->colour='r';

rotate\_right(g\_par);

}

}

}

root->colour='b';

}

void print\_tree(node \*root)

{

if(root->left!='\0')

print\_tree(root->left);

cout<<root->info<<" "<<root->colour<<"\n";

if(root->right!='\0')

print\_tree(root->right);

}

int main()

{

int i,choice,temp1,temp2,hash;

node \*p1,\*p2;

cout<<"press 1 for insert element\npress 2 for searching element\n";

cout<<"press 4 for print tree\npress 5 for exit\nenter choice\n";

cin>>choice;

while(choice!=5)

{

switch(choice)

{

case 1:

cout<<"enter number\n";

cin>>temp1;

p1=new node;

p1->info=temp1;

p1->left=p1->right=p1->parent='\0';

p1->colour='r';

insert\_node(root,p1);

set\_node(root,p1);

break;

case 2:

cout<<"enter no. to search\n";

cin>>temp1;

p1=root;

while(p1!='\0'&&p1->info!=temp1)

{

if(p1->info<temp1)

p1=p1->right;

if(p1->info>temp1)

p1=p1->left;

}

if(p1=='\0')

cout<<temp1<<" is not present in the tree\n";

else

cout<<p1->info<<" "<<p1->colour<<" "<<p1->parent<<"\n";

break;

case 4:

print\_tree(root);

cout<<"\n";

break;

default:

cout<<"invalid choice\n";

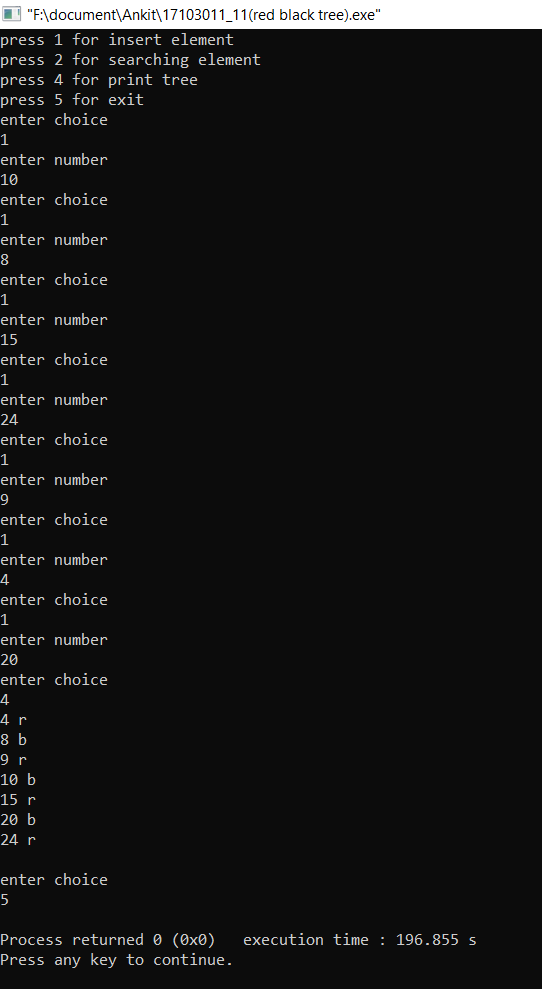
}

cout<<"enter choice\n";

cin>>choice;

}

}



**PROGRAM NO.12**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT LONGEST COMMON SUBSEQUENCE**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 13/03/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

int lcs(string s1,string s2,int length1,int length2)

{

int m[length1+1][length2+1];

int i,j;

for(i=0;i<=length1;i++)

{

for(j=0;j<=length2;j++)

{

if(i==0||j==0)

m[i][j]=0;

else if(s1[i-1]==s2[j-1])

m[i][j]=1+m[i-1][j-1];

else

m[i][j]=max(m[i-1][j],m[i][j-1]);

}

}

return m[length1][length2];

}

int main()

{

string s1,s2;

cout<<"enter string 1\n";

cin>>s1;

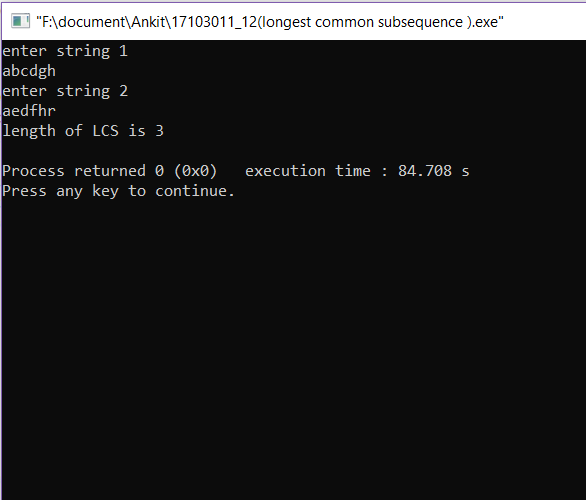
cout<<"enter string 2\n";

cin>>s2;

int a=lcs(s1,s2,s1.length(),s2.length());

cout<<"length of LCS is "<<a<<"\n";

}



**PROGRAM NO.13**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT MATRIX CHAIN MULTIPLICATION**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 03/04/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

int mcm(int array[],int size)

{

int m[size][size];

int i,j,k,cost;

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

m[i][i]=0;

for(int l=2;l<size;l++)

{

for(i=1;i<size-l+1;i++)

{

j=i+l-1;

m[i][j]=INT\_MAX;

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

{

cost=m[i][k]+m[k+1][j]+array[i-1]\*array[k]\*array[j];

if(cost<m[i][j])

m[i][j]=cost;

}

}

}

return m[1][size-1];

}

int main()

{

int n;

cout<<"enter no. of matrixes\n";

cin>>n;

int a[n+1];

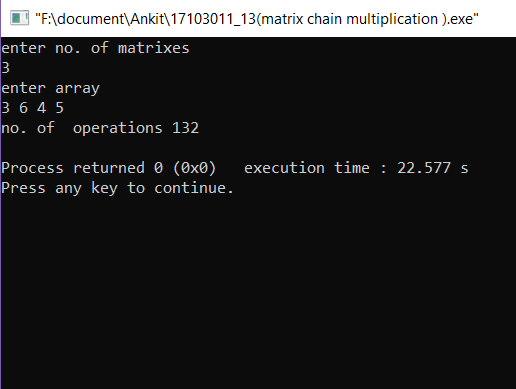
cout<<"enter array\n";

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

cin>>a[i];

cout<<"no. of operations "<<mcm(a,n+1)<<"\n";

}



**PROGRAM NO.14**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT FRACTIONAL KNAPSACK PROBLEM**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 10/04/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

struct node

{

float ratio,value,weight;

};

bool compare(node n1,node n2)

{

return n1.ratio>n2.ratio;

}

int fractional\_knapsack(node array[],int w,int size)

{

sort(array,array+size,compare);

int i=0,total\_value=0,current\_w=0;

while(i<size&&(current\_w+array[i].weight)<=w)

{

current\_w+=array[i].weight;

total\_value+=array[i].value;

i++;

}

if(i<size&&w-current\_w>0)

total\_value+=(w-current\_w)\*array[i].ratio;

return total\_value;

}

int main()

{

int n,w;

cout<<"enter no. of values and weight of knapsack\n";

cin>>n>>w;

node array[n];

cout<<"enter values and weights\n";

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

{

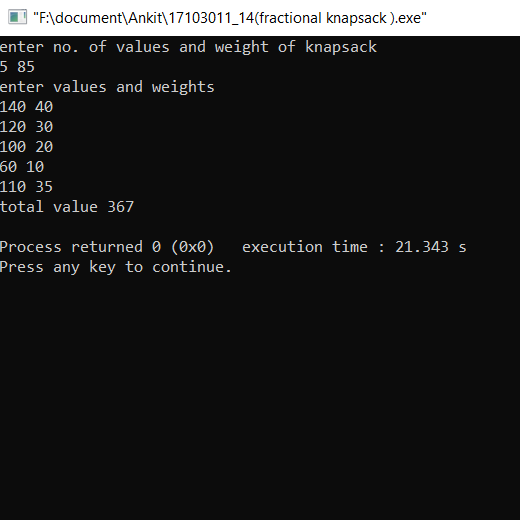
cin>>array[i].value>>array[i].weight;

array[i].ratio=array[i].value/array[i].weight;

}

cout<<"total value "<<fractional\_knapsack(array,w,n)<<"\n";

}



**PROGRAM NO.15**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT 0-1 KNAPSACK PROBLEM**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 10/04/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

int knapsack(int value[],int weight[],int w,int size)

{

int i,j,k,m[size+1][w+1];

for(i=0;i<size+1;i++)

{

for(j=0;j<w+1;j++)

{

if(i==0||j==0)

m[i][j]=0;

else if(weight[i-1]>j)

m[i][j]=m[i-1][j];

else

m[i][j]=max(m[i-1][j],m[i-1][j-weight[i-1]]+value[i-1]);

}

}

return m[size][w];

}

int main()

{

int n,w;

cout<<"enter no. of values and weight of knapsack\n";

cin>>n>>w;

int value[n],weight[n];

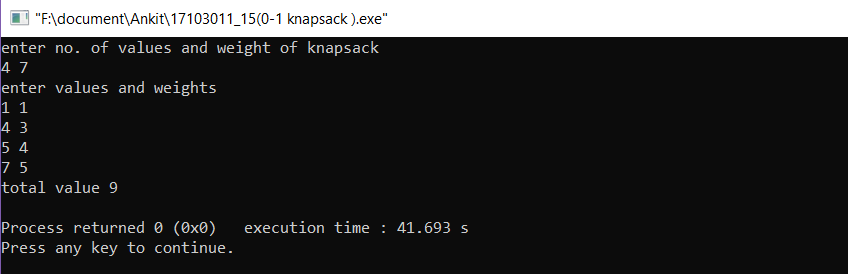
cout<<"enter values and weights\n";

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

cin>>value[i]>>weight[i];

cout<<"total value "<<knapsack (value ,weight ,w ,n)<<"\n";

}



**PROGRAM NO.16**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO IMPLEMENT DEPTH FIRST SEARCH AND BREADTH FIRST SEARCH IN GRAPH**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 24/04/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

struct node

{

int value,status;

};

void dfs(node array[],int \*\*matrix,int n,int edges)

{

int i=0,j,k;

node temp;

stack <node> s;

while(i<n)

{

if(array[i].status==0)

{

s.push(array[i]);

while(!s.empty())

{

temp=s.top();

s.pop();

j=0;

while(j<n)

{

if(temp.value==array[j].value)

break;

j++;

}

array[j].status=2;

cout<<array[j].value<<" ";

k=0;

while(k<n)

{

if(matrix[j][k]==1&&array[k].status==0)

{

array[k].status=1;

s.push(array[k]);

}

k++;

}

}

}

i++;

}

}

void bfs(node array[],int \*\*matrix,int n,int edges)

{

int i=0,j,k;

node temp;

queue <node> q;

while(i<n)

{

if(array[i].status==0)

{

q.push(array[i]);

while(!q.empty())

{

temp=q.front();

q.pop();

j=0;

while(j<n)

{

if(temp.value==array[j].value)

break;

j++;

}

array[j].status=2;

cout<<array[j].value<<" ";

k=0;

while(k<n)

{

if(matrix[j][k]==1&&array[k].status==0)

{

array[k].status=1;

q.push(array[k]);

}

k++;

}

}

}

i++;

}

}

int main()

{

int n,edges;

cout<<"enter no. of nodes and edges\n";

cin>>n>>edges;

cout<<"enter nodes\n";

node array[n];

int i=0,j;

while(i<n)

{

cin>>array[i].value;

array[i].status=0;

i++;

}

cout<<"enter edges\n";

int \*\*matrix=(int\*\*)malloc(sizeof(int\*)\*n);

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

matrix[i]=(int\*)malloc(sizeof(int)\*n);

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

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

matrix[i][j]=0;

int temp1,temp2;

i=0;

while(i<edges)

{

cin>>temp1>>temp2;

matrix[temp1][temp2]=1;

i++;

}

cout<<"DFS = ";

dfs(array,matrix,n,edges);

cout<<"\n";

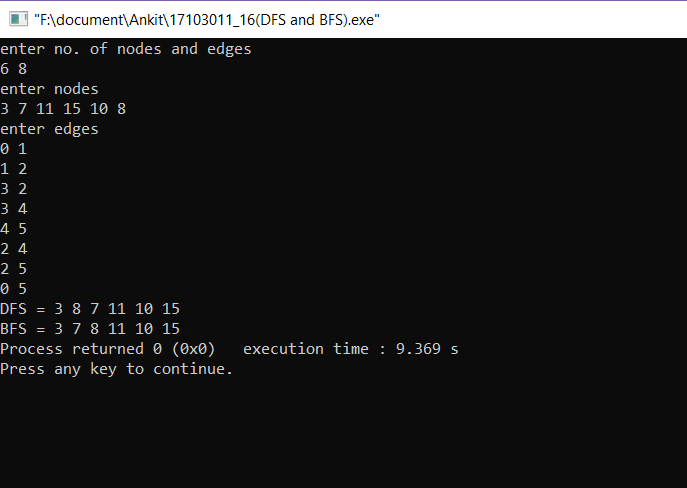
for(j=0;j<n;j++)

array[j].status=0;

cout<<"BFS = ";

bfs(array,matrix,n,edges);

}



**PROGRAM NO.17**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO FIND MINIMUM SPANNING TREE USING PRIM'S ALGORITHM**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 24/04/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

int find\_min(int value[],int flag[],int n)

{

int i=0,min=INT\_MAX,index=-1;

while(i<n)

{

if(flag[i]==0&&value[i]<min)

{

min=value[i];

index=i;

}

i++;

}

return index;

}

void prim(int\*\* matrix,int n)

{

int value[n],par[n],flag[n],count=0,i=0,index;

while(i<n)

{

value[i]=INT\_MAX;

par[i]=-1;

flag[i]=0;

i++;

}

value[0]=0;

while(count<n-1)

{

index=find\_min(value,flag,n);

flag[index]=1;

i=0;

while(i<n)

{

if(matrix[index][i]>0&&flag[i]==0&&matrix[index][i]<value[i])

{

par[i]=index;

value[i]=matrix[index][i];

}

i++;

}

count++;

}

cout<<"included edges are\n";

i=1;

while(i<n)

{

cout<<i<<"---"<<par[i]<<"\n";

i++;

}

}

int main()

{

int n,edges,i=0,j;

cout<<"enter no. of nodes and edges\n";

cin>>n>>edges;

cout<<"enter edge and weight of edge\n";

int \*\*matrix=(int\*\*)malloc(sizeof(int\*)\*n);

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

matrix[i]=(int\*)malloc(sizeof(int)\*n);

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

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

matrix[i][j]=0;

int temp1,temp2,temp3;

i=0;

while(i<edges)

{

cin>>temp1>>temp2>>temp3;

matrix[temp1][temp2]=temp3;

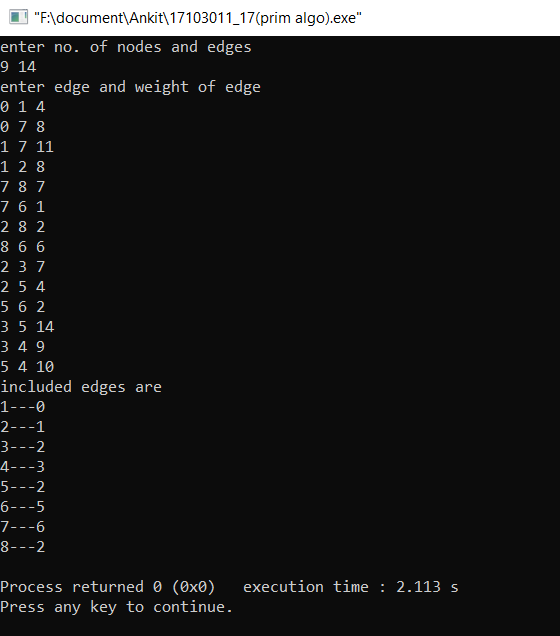
matrix[temp2][temp1]=temp3;

i++;

}

prim(matrix,n);

}



**PROGRAM NO.18**

**/\***

**PROGRAM NAME = WRITE A PROGRAM TO FIND MINIMUM SPANNING TREE USING KRUSKAL'S ALGORITHM**

**ROLL NO. = 17103011**

**NAME = ANKIT GOYAL**

**DATE = 24/04/2019 \*/**

#include<bits/stdc++.h>

using namespace std;

struct edge

{

int s,d,value;

};

bool compare(edge e1,edge e2)

{

if(e1.value>e2.value)

return 0;

return 1;

}

int find\_parent(int par[],int temp,int n)

{

while(par[temp]!=-1)

temp=par[temp];

return temp;

}

void kruskal\_algo(edge array[],int n,int edges)

{

int par[n],count=0,i=0,index,set1,set2,temp3,edge\_count=0;

sort(array,array+edges,compare);

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

par[i]=-1;

queue <int> q;

while(count<n-1)

{

set1=find\_parent(par,array[edge\_count].s,n);

set2=find\_parent(par,array[edge\_count].d,n);

if(set1!=set2)

{

count++;

q.push(edge\_count);

par[set1]=set2;

}

edge\_count++;

}

cout<<"included edges are\nn1---n2 weight\n ";

while(q.empty()!=1)

{

i=q.front();

cout<<array[i].s<<"---"<<array[i].d<<" "<<array[i].value<<"\n ";

q.pop();

}

}

int main()

{

int n,edges,i=0,j;

cout<<"enter no. of nodes and edges\n";

cin>>n>>edges;

cout<<"enter edge and weight of edge\n";

edge array[edges];

i=0;

while(i<edges)

{

cin>> array[i].s>>array[i].d>>array[i].value;

i++;

}

kruskal\_algo(array,n,edges);

}

