//take the value into 2d array and display the value of 2d array

int main()

{

int disp[2][3];

int i, j;

for(i=0; i<2; i++)

{

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

{

printf("Enter value for disp[%d][%d]:", i, j);

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

}

}

printf("\n\nTwo Dimensional array elements:\n");

for(i=0; i<2; i++)

{

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

{

printf("%2d ", disp[i][j]);

if(j==2)

{

printf("\n");

}

}

}

return 0;

}

//Take an array of n number and print out the even values of the array

#include <stdio.h>

void main()

{

int array[100], i, num;

printf("Enter the size of an array :\n");

scanf("%d", &num);

printf("Enter the elements of the array :\n");

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

{

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

}

printf("\nEven numbers in the array are :- ");

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

{

if (array[i] % 2 == 0)

{

printf("%d \t", array[i]);

}

}

}

//Take an array of n number and print out the values in even in every index

#include <stdio.h>

void main()

{

int array[100], i, num;

printf("Enter the size of an array :\n");

scanf("%d", &num);

printf("Enter the elements of the array :\n");

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

{

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

}

printf("\nValues in even in every index of the array are :- \n");

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

{

if (i % 2 == 0)

{

printf("%d \t", array[i]);

}

}

}

//Adjacency Matrix

#include<stdio.h>

//#include<iostream>

int adj[100][100];

int main()

{

int node,edge;

int n1,n2,i,j;

printf("Enter the number of node: ");

scanf("%d",&node);

printf("Enter the number of edge: ");

scanf("%d",&edge);

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

{

printf("enter row :");

scanf("%d",&n1);

printf("enter colomn :");

scanf("%d",&n2);

adj[n1][n2]=1;

adj[n2][n1]=1;

}

printf("\nAdjacency Matrix: \n\n");

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

{

printf("|");

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

{

printf("%d\t",adj[i][j]);

}

printf("|\n");

}

return 0;

}

//DFS Code

#include<stdio.h>

int adj[50][50],visit[50];

int edge,node;

int i,j,n1,n2,s;

dfs(int i)

{

//int i;

visit[i]=1;

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

if(adj[i][j] ==1 && visit[j] == 0)

{

printf("%d->%d , ",i,j);

dfs(j);

}

}

int main()

{

printf("enter the number of nodes:");

scanf("%d",&node);

printf("enter the number of edges:");

scanf("%d",&edge);

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

{

printf("enter row :");

scanf("%d",&n1);

printf("enter colomn :");

scanf("%d",&n2);

adj[n1][n2]=1;

adj[n2][n1]=1;

}

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

{

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

{

printf("%d\t",adj[i][j]);

}

printf("\n");

}

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

{

visit[i]=0;

}

printf("\nEnter the starting node :");

scanf("%d", &s);

printf("T = {");

i = s;

while(visit[i]==0)

{

dfs(i);

}i++;

printf(" }\n\n");

//return 0;

}

//BFS Code

//Insertion sort in c program

#include <stdio.h>

int main()

{

int array[100], n, j, k, ptr, temp;

printf("Enter number of elements : ");

scanf("%d", &n);

printf("Enter %d integers :\n", n);

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

scanf("%d", &array[k]);

array[0] = -99999999;

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

{

temp = array[k];

for ( ptr = k-1; temp < array[ptr]; ptr = ptr-1)

{

array[ptr+1] = array[ptr];

}

array[ptr+1] = temp;

}

printf("Sorted list in ascending order:\n");

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

printf("%d\n", array[k]);

return 0;

}

//Selection sort in c p;rogram

#include <stdio.h>

int main()

{

int array[100], n, j, k, min, LOC, temp;

printf("Enter number of elements : ");

scanf("%d", &n);

printf("Enter %d integers :\n", n);

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

scanf("%d", &array[k]);

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

{

min = array[k];

LOC = k;

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

{

if (min > array[j])

{

min = array[j];

LOC = j;

}

}

temp = array[k];

array[k] = array[LOC];

array[LOC] = temp;

}

printf("Sorted list in ascending order:\n");

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

printf("%d\n", array[k]);

return 0;

}

//Bubble sort in c program

#include <stdio.h>

int main()

{

int array[100], n, k, ptr, swap;

printf("Enter number of elements : ");

scanf("%d", &n);

printf("Enter %d integers :\n", n);

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

scanf("%d", &array[k]);

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

{

for (ptr = 0 ; ptr < n - k - 1; ptr++)

{

if (array[ptr] > array[ptr+1])

{

swap = array[ptr];

array[ptr] = array[ptr+1];

array[ptr+1] = swap;

}

}

}

printf("Sorted list in ascending order:\n");

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

printf("%d\n", array[k]);

return 0;

}

//Quick sort in c program

#include<stdio.h>

int array[25],start,end;

int partition(int start,int end)

{

int pivot,i,pi,temp;

pivot = array[end];

pi = start;

for(i=start; i<=end-1 ; i++)

{

if(array[i]<=pivot)

{

temp=array[i];

array[i]=array[pi];

array[pi]=temp;

pi = pi+1;

}

}

temp=array[end];

array[end]=array[pi];

array[pi]=temp;

return pi;

}

void quicksort(int start,int end)

{

int pi;

if(start<end)

{

pi = partition(start,end);

quicksort(start,pi-1);

quicksort(pi+1,end);

}

}

int main()

{

int i, count;

printf("How many elements are u going to enter : ");

scanf("%d",&count);

printf("Enter %d elements: \n", count);

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

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

start = 0;

end = count-1;

quicksort(start,end);

printf("Order of Sorted elements: \n");

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

printf(" %d",array[i]);

return 0;

}

//Heap Sort in c program

#include <stdio.h>

#include <stdlib.h>

int a[20], i, n, heap\_size;

void max\_heap()

{

heap\_size=n; //a.length=n

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

{

max\_heapify(i);

}

}

void max\_heapify(int j)

{

int l=2\*j;

int r=2\*j+1;

int largest=j;

int e;

if(l<=heap\_size&&a[l]>a[largest])

{

largest=l;

}

if(r<=heap\_size&&a[r]>a[largest])

{

largest=r;

}

if(largest!=j)

{

e=a[j];

a[j]=a[largest];

a[largest]=e;

max\_heapify(largest);

}

}

void heapsort()

{

int c;

heap\_size=n;

max\_heap();

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

{

c=a[i];

a[i]=a[1];

a[1]=c;

heap\_size=heap\_size-1;

max\_heapify(1);

}

}

main()

{

int p, q;

printf("array size:");

scanf("%d",&n);

printf("Insert values: ");

printf("\n");

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

{

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

}

printf("\n");

heapsort();

printf("Sorted array :\n");

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

{

printf("%d\t",a[q]);

}

printf("\n");

}

//BMF code

#include <stdio.h>

#include <stdlib.h>

//array is starting from 0 index.

int a[10000][10000], dist[10000], prev[10000];

int vertex, edge;

int min(int i, int j)

{

if(i<j)

return i;

else

return j;

}

void update(int i, int j)

{

dist[j] = min(dist[j], dist[i]+a[i][j]);

}

int main()

{

freopen("bmf.txt", "r", stdin);

int m, i, j, k, n, w8, s;

printf("Enter total vertex: \n");

scanf("%d", &vertex);

printf("Enter total edge: \n");

scanf("%d", &edge);

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

{

dist[i]=999999;

}

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

{

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

{

a[i][j]=0;

}

}

printf("Enter edges and weight: \n");

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

{

scanf("%d %d %d", &m, &n, &w8);

a[m][n] = w8;

}

printf("Enter Source : \n\n");

scanf("%d",&s);

dist[s] = 0;

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

{

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

{

for(int k = 0; k< vertex; k++)

{

if(a[j][k]!=0)

update(j, k);

}

}

}

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

{

printf("Dist(%d) = %d\n", i, dist[i]);

}

return 0;

}