# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 8

**1)** **Demonstrate the** Implementation of addition and deletion of edges in a directed graph using adjacency matrix.

CODE

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

*int* adj[MAX][MAX];

*int* n;

*void* create\_graph();

*void* display();

*void* insert\_edge(*int* *origin*, *int* *destin*);

*void* del\_edge(*int* *origin*, *int* *destin*);

*int* main()

{

*int* choice, origin, destin;

    create\_graph();

    while (1)

    {

        printf("\n1.Insert an edge\n");

        printf("2.Delete an edge\n");

        printf("3.Display\n");

        printf("4.Exit\n");

        printf("\nEnter your choice : ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("\nEnter an edge to be inserted : ");

            scanf("%d %d", &origin, &destin);

            insert\_edge(origin, destin);

            break;

        case 2:

            printf("\nEnter an edge to be deleted : ");

            scanf("%d %d", &origin, &destin);

            del\_edge(origin, destin);

            break;

        case 3:

            display();

            break;

        case 4:

            exit(1);

        default:

            printf("\nWrong choice\n");

            break;

        }

    }

    return 0;

}

*void* create\_graph()

{

*int* i, max\_edges, origin, destin;

    printf("\nEnter number of vertices : ");

    scanf("%d", &n);

    max\_edges = n \* (n - 1);

    for (i = 1; i <= max\_edges; i++)

    {

        printf("\nEnter edge %d(-1 -1) to quit : ", i);

        scanf("%d %d", &origin, &destin);

        if ((origin == -1) && (destin == -1))

            break;

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

        {

            printf("\nInvalid edge!\n");

            i--;

        }

        else

            adj[origin][destin] = 1;

    }

}

*void* insert\_edge(*int* *origin*, *int* *destin*)

{

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

    {

        printf("\nOrigin vertex does not exist\n");

        return;

    }

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

    {

        printf("\nDestination vertex does not exist\n");

        return;

    }

    adj[*origin*][*destin*] = 1;

}

*void* del\_edge(*int* *origin*, *int* *destin*)

{

    if (*origin* < 0 || *origin* >= n || *destin* < 0 || *destin* >= n || adj[*origin*][*destin*] == 0)

    {

        printf("\nThis edge does not exist\n");

        return;

    }

    adj[*origin*][*destin*] = 0;

}

*void* display()

{

*int* i, j;

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

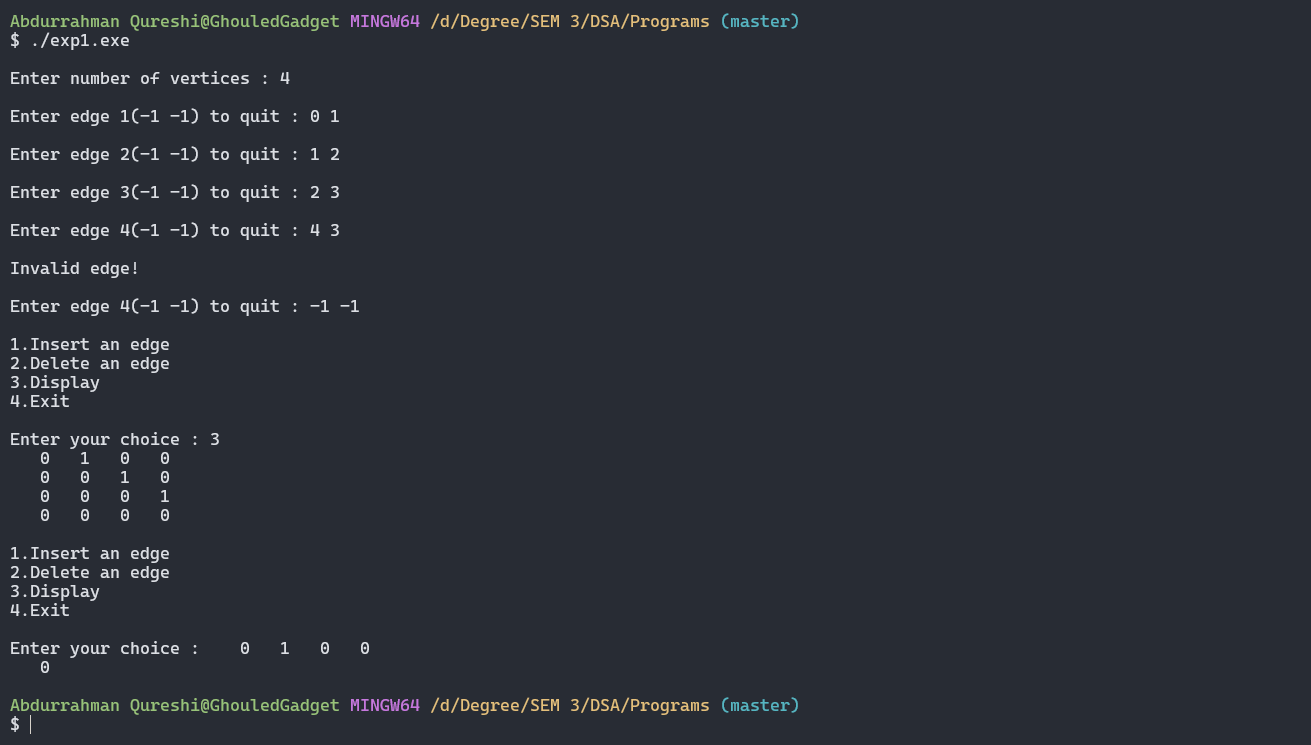
            printf("%4d", adj[i][j]);

        printf("\n");

    }

}

OUTPUT



Tools used :

Software: Dev c++

Hardware: Lab Computers

References: Mam notes.

Conclusion

* An Adjacency matrix is a square matrix used to represent a finite graph.
* It contains the information about the edges and its cost.
* If the value at the Ith row and Jth column are zero, it means an edge does not exist between these two vertices. Else you got the edge and cost of that edge.