**Code:-**

# **Name : Rajkumar B L**

# **Reg.No : 2047120**

# **Course : MCS 271 Data Structure (Lab 11 – Prim & Kruskal)**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* Name : Rajkumar B L

 \* Reg  : 2047120

 \* Lab  : 11

 \* Program : Kruskal & Prim

 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

int find(int);

int uni(int, int);

void prim();

void kruskal();

int i, j, k, a, b, u, v, n, ne = 1;

int min, mincost = 0, cost[9][9], parent[9];

void main()

{

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\*  Name : Rajkumar B L     \*\n\*  Reg  : 2047120          \*\n\*  Lab  : 11               \*\n\*  Prg  : Kruskal & Prim   \*\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

    int ch;

    do

    {

        printf("\n=========================\n\tMenu\n=========================\n");

        printf("1. Kruskal's Algorithm\n");

        printf("2. Prim's Algorithm\n");

        printf("3. Exit\n");

        printf("=========================\n");

        printf("Enter your choice: ");

        fflush(stdin);

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            kruskal();

            break;

        case 2:

            prim();

            break;

        case 3:

            printf("Bye.\n");

            break;

        default:

            printf("Invalid choice.\n");

            break;

        }

        printf("\n");

    } while (ch != 3);

}

void prim()

{

    int a, b, u, v, n, i, j, ne = 1;

    int visited[10] = {0}, min, mincost = 0, cost[10][10];

    printf("\nImplementation of Prim's algorithm\n");

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

    scanf("%d", &n);

    printf("Enter the adjacency matrix: \n");

    for (i = 1; i <= n; i++)

        for (j = 1; j <= n; j++)

        {

            scanf("%d", &cost[i][j]);

            if (cost[i][j] == 0)

                cost[i][j] = 999;

        }

    visited[1] = 1;

    printf("\nThe edges of Minimum Cost Spanning Tree are:-");

    while (ne < n)

    {

        for (i = 1, min = 999; i <= n; i++)

            for (j = 1; j <= n; j++)

                if (cost[i][j] < min)

                    if (visited[i] != 0)

                    {

                        min = cost[i][j];

                        a = u = i;

                        b = v = j;

                    }

        if (visited[u] == 0 || visited[v] == 0)

        {

            printf("\nEdge %d:(%d %d) cost:%d", ne++, a, b, min);

            mincost += min;

            visited[b] = 1;

        }

        cost[a][b] = cost[b][a] = 999;

    }

    printf("\nMinimun cost = %d!", mincost);

    //getch();

}

void kruskal()

{

    printf("\nImplementation of Kruskal's algorithm\n");

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

    scanf("%d", &n);

    printf("Enter the adjacency matrix: \n");

    for (i = 1; i <= n; i++)

    {

        for (j = 1; j <= n; j++)

        {

            scanf("%d", &cost[i][j]);

            if (cost[i][j] == 0)

                cost[i][j] = 999;

        }

    }

    printf("\nThe edges of Minimum Cost Spanning Tree are:-");

    while (ne < n)

    {

        for (i = 1, min = 999; i <= n; i++)

        {

            for (j = 1; j <= n; j++)

            {

                if (cost[i][j] < min)

                {

                    min = cost[i][j];

                    a = u = i;

                    b = v = j;

                }

            }

        }

        u = find(u);

        v = find(v);

        if (uni(u, v))

        {

            printf("%d edge (%d,%d) = %d\n", ne++, a, b, min);

            mincost += min;

        }

        cost[a][b] = cost[b][a] = 999;

    }

    printf("\nMinimum cost = %d!\n", mincost);

    //getch();

}

int find(int i)

{

    while (parent[i])

        i = parent[i];

    return i;

}

int uni(int i, int j)

{

    if (i != j)

    {

        parent[j] = i;

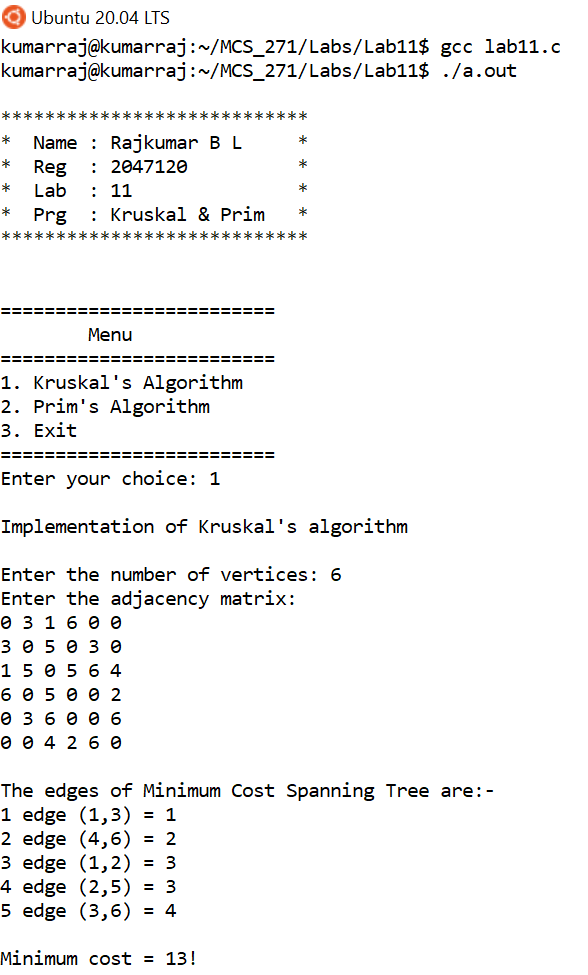
        return 1;

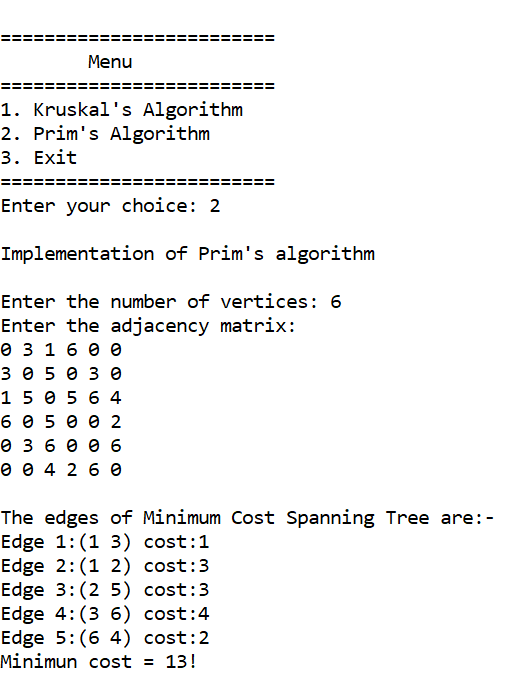
    }

    return 0;

}

**Output:**

****

****