```
#include<stdio.h>
#define SIZE 20
#define infinity 999


void read_graph(int *nv, int adj[][SIZE])
{
    int i, j;

    printf("\nEnter the number of vertices : ");
    scanf("%d", nv);

    printf("\nEnter the adjecency matrix (order %d x %d) :\n", *nv, *nv);
    for( i = 1; i <= *nv; i++ )
        for( j = 1; j <= *nv; j++ )
            scanf("%d", &adj[i][j]);
}


int find(int i, int parent[])
{
        while(parent[i])
            i = parent[i];
        return i;
}

int uni(int i,int j, int parent[])
{
        if(i != j)
        {
                parent[j] = i;
                return 1;
        }
        return 0;
}

void Kruskal(int adj[][SIZE], int *nv)
{
    int i, j, a, b, u, v, ne=1;
    int min,mincost = 0;
    int parent[SIZE] = {0};
    int adj_temp[SIZE][SIZE];

    if( !*nv )
        {
            printf("\nPlease read a graph...\n");
            return;
        }

    for( i = 1; i <= *nv; i++ )
```

```
        for( j = 1; j <= *nv; j++ )
        {
            adj_temp[i][j] = adj[i][j];
            if(adj_temp[i][j] == 0)
                                adj_temp[i][j] = infinity;
        }
        printf("The edges of Minimum Cost Spanning Tree are\n");
        while(ne < *nv)
        {
                for(i = 1, min = infinity; i <= *nv; i++)
                {
                        for(j = 1; j <= *nv; j++)
                        {
                                if(adj_temp[i][j] < min)
                                {
                                        min = adj_temp[i][j];
                                        a = u = i;
                                        b = v = j;
                                }
                        }
                }
                u = find(u, parent);
                v = find(v, parent);
                if(uni(u, v, parent))
                {
                        printf("%d edge (%d,%d) = %d\n", ne++, a, b, min);
                        mincost += min;
                }
                adj_temp[a][b] = adj_temp[b][a] = infinity;
        }
        printf("\nSuccessfully created a spanning tree and its minimum cost is %d
\n", mincost);

}

void display( int adj[][SIZE], int *nv, int flag)
{
    int i, j;

    if( !*nv )
        {
            printf("\nPlease read a graph...\n");
            return;
        }
    printf("\nThe given graph (adjacency matrix) is:\n");
        for( i = 1; i <= *nv; i++ )
        {
                for( j = 1; j <= *nv; j++ )
                        printf("%d ", adj[i][j] );
                printf("\n");
        }

        if( flag )
            Kruskal( adj, nv );

}


int main()
{
        int adj[SIZE][SIZE];
        int nv;
        int flag = 0;
        int e = 1, ch;

        while( e )
        {
                printf( "\n-------------MENU-------------\n" );
                printf( "\n\t1. Read Graph\n\t2. Display\n\t3. Kruskal's Algorithm
- Spanning Tree\n\t4. Exit\n" );
                printf( "\n-----------------------------\n" );
                printf( "\n Enter your choice:" );
                scanf( "%d", &ch );

                switch( ch )
                {
                        case 1: read_graph( &nv, adj );
                             break;
                        case 2: display( adj, &nv, flag);
                                 break;
                        case 3: flag = 1;
                                Kruskal( adj, &nv );
                             break;
                        case 4 : e = 0;
                                 break;
                        default: printf( "\n Invalid choice \n" );
                }

        }
        return 0;
}
```