```c
#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 = 0; i < *nv; i++ )
        for( j = 0; j < *nv; j++ )
            scanf("%d", &adj[i][j]);
}


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

    if( !*nv )
        {
            printf("\nPlease read a graph...\n");
            return;
        }
    printf("\nThe given graph (adjacency matrix) is:\n");
        for( i = 0; i < *nv; i++ )
        {
                for( j = 0; j < *nv; j++ )
                        printf("%d ", adj[i][j] );
                printf("\n");
        }

        if( flag )
        {
            printf("\nSpanning Tree is: \n");
            for( i = 0; i < *nv; i++ )
            {
                    for( j = 0; j < *nv; j++ )
                            printf("%d ", st[i][j] );
                    printf("\n");
            }
            printf("\nThe minimum cost is %d ", cost);
        }
}
```

```
int Prims(int adj[][SIZE], int st[][SIZE], int *nv)
{
        int cost[SIZE][SIZE];
        int u,v,min_distance,distance[SIZE],from[SIZE];
        int visited[SIZE],no_of_edges,i,min_cost,j;

        if( !*nv )
        {
            printf("\nPlease read a graph...\n");
            return 0;
        }

        for(i = 0; i < *nv; i++ )
                for( j = 0; j < *nv; j++ )
                {
                        if( adj[i][j] == 0 )
                                cost[i][j] = infinity;
                        else
                                cost[i][j] = adj[i][j];
                        st[i][j] = 0;
                }

        distance[0] = 0;
        visited[0] = 1;

        for( i = 1; i < *nv; i++ )
        {
                distance[i] = cost[0][i];
                from[i] = 0;
                visited[i] = 0;
        }

        min_cost = 0;
        no_of_edges = *nv - 1;

        while( no_of_edges > 0 )
        {

                min_distance = infinity;
                for( i = 1; i < *nv; i++ )
                        if( visited[i] == 0 && distance[i] < min_distance )
                        {
                                v = i;
                                min_distance = distance[i];
                        }

                u = from[v];

                st[u][v] = distance[v];
                st[v][u] = distance[v];
```

```
                no_of_edges--;
                visited[v] = 1;


                for( i = 1; i < *nv; i++ )
                        if(visited[i] == 0 && cost[i][v] < distance[i] )
                        {
                                distance[i] = cost[i][v];
                                from[i] = v;
                        }

                min_cost = min_cost + cost[u][v];
        }

        return( min_cost );
}

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

        while( e )
        {
                printf( "\n--------------MENU--------------\n" );
                printf( "\n\t1. Read Graph\n\t2. Display\n\t3. Prim'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, st, &nv, flag, cost );
                                 break;
                        case 3: flag = 1;
                                cost = Prims( adj, st, &nv );
                                if( cost )
                                    printf("\nSuccessfully created a spanning tree
and its minimum cost is %d \n", cost );
                             break;
                        case 4 : e = 0;
                                 break;
                        default: printf( "\n Invalid choice \n" );
                }
```

```
        }
        return 0;
}
```