```
#include<iostream>
using namespace std;

const int MAXNODES = 10;
void fnPrims(int n, int cost[MAXNODES][MAXNODES]);
void fnGetMatrix(int n,int a[MAXNODES][MAXNODES]);

/
********************************************************************
**********
*Function   : main
*Input parameters:
*   int argc - no of commamd line arguments
*   char **argv - vector to store command line argumennts
*RETURNS    :   0 on success
********************************************************************
**********/
int main( int argc, char **argv)

{
    int a[MAXNODES][MAXNODES] = {{0, 3, 9999, 7, 9999},
                {3, 0, 4, 2, 9999},
                {9999, 4, 0, 5, 6},
                {7, 2, 5, 0, 4},
                {9999, 9999, 6, 4, 0}};

    int n = 5;

    cout << "Enter the number of vertices : ";
    cin >> n;

    fnGetMatrix(n,a);
    fnPrims(n,a);

    return 0;
}

/
********************************************************************
**********
*Function   : fnPrims
*Description    : Function to find Minimum Cost Spanning Tree of a
given
*               undirected graph using Prims algorithm.
*Input parameters:
*   int n   - no of vertices in the graph
*   int cost[][] - cost adjacency matrix of the graph
*RETURNS    : no value
********************************************************************
**********/
void fnPrims(int n, int cost[MAXNODES][MAXNODES])
{
    int i, j, u, v, min;
    int sum, k, t[MAXNODES][2], p[MAXNODES], d[MAXNODES],
s[MAXNODES];
    int source, count;

    min = 9999;
    source = 0;

    for(i=0; i<n; i++)  //finding the node with minimum cost
    {
        for(j=0; j<n; j++)
        {
            if(cost[i][j] != 0 && cost[i][j] <= min)
            {
                min = cost[i][j];
                source = i;
            }
        }
    }

    for(i=0; i<n; i++)
    {
        d[i] = cost[source][i]; //initializing the array with th
cost of all the nodes from source.
        s[i] = 0;
        p[i] = source;
    }
    s[source] = 1;
    sum = 0;
    k = 0;
    count = 0;

    while (count != n-1)
    {
        min = 9999;
        u = -1;
        for(j=0; j<n; j++)
        {
            if(s[j] == 0)
            {
                if(d[j] <= min)
                {
                     min = d[j];
                     u = j;
                }
            }
        }

        t[k][0] = u;
        t[k][1] = p[u];
        k++;
        count++;
        sum += cost[u][p[u]];
        s[u] = 1;
        for(v=0; v<n; v++)
        {
            if(s[v]==0 && cost[u][v]<d[v])
            {
                d[v] = cost[u][v];
                p[v] = u;
            }
        }
    }

    if(sum >= 9999)
        cout << "\nSpanning tree does not exist\n";
    else
    {
        cout << "\nThe spanning tree exists and minimum cost
spanning tree is \n";
        for(i=0; i<k; i++)
            cout << t[i][1] << " " << t[i][0] << endl;

        cout << "\nThe cost of the minimum cost spanning tree is "
<< sum << endl;
    }
}

/
*******************************************************************
**********
*Function   : fnGetMatrix
*Description    : Function to read cost adjacency matrix of the
graph
*Input parameters:
*   int n   - no of vertices in the graph
*   int a[][] - cost adjacency matrix of the graph
*RETURNS    : no value
*******************************************************************
**********/
void fnGetMatrix(int n,int a[MAXNODES][MAXNODES])
{
    int i, j;

    cout << "Enter the Cost Adjacency Matrix" << endl;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            cin >> a[i][j];
}
```

Output Sample 2:

```
Enter the number of vertices : 5 Enter the Cost Adjacency Matrix

0 3 9999 7 9999

3 0 4 2 9999
```

```
9999 4 0 5 6

7 2 5 0 4

9999 9999 6 4 0

The spanning tree exists and minimum cost spanning tree is

3 1

1 0

3 4

1 2

The cost of the minimum cost spanning tree is 13
```