```cpp
//*****************************************************************//
//**************string  Author = "SACHIN SAINI " ******************//
//*****************************************************************//


#include<bits/stdc++.h>
using namespace std;

class Graph
{
public:
    int rows=6,columns=6;
    int adjMatrix[6][6]; //To represent the graph
    bool VisitedArray[6]; //to store the visited node
    int parentArray[6]={-1,-1,-1,-1,-1,-1}; //store the constructed MST
    int keys[6]; //used to store key values of all vertices
public:
    void initializeMatrix(); //to initialize the adjMatrix
    void addEdge(int ,int ,int); //Add an edge to the graph
    void printAdjMatrix();// To print adjMatrix
    int findMinimum(); //to find minimum among all the nodes adjacent to MST
    void assignKeyValues();//Initially initialize key values
};

Graph graph1;//create an object of graph type

void Graph::initializeMatrix()
{
    int i,j;
    for(i=0;i<rows;i++)
    {
        for(j=0;j<columns;j++)
        {
            adjMatrix[i][j]=0;
        }
    }
}

void Graph::addEdge(int source,int dest,int weight)
{
    adjMatrix[source][dest]=weight;// add an from source to destination
    adjMatrix[dest][source]=weight; //as undirected graph so add opposite edge
}

void Graph::printAdjMatrix()
{
    cout<<"-------------------------------------"<<endl;
    for(int i=0;i<rows;i++)
    {
        for(int j=0;j<columns;j++)
        {
            cout<<adjMatrix[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<"-------------------------------------"<<endl;
}

//find Minimum edge
int Graph::findMinimum()
{
    int min = INT_MAX,minIndex;
    for(int i=0;i<rows;i++)
    {
         if(VisitedArray[i]==false && keys[i]<min) //if node is not visited and key value of that node is less then the minimum value
         {
             min= keys[i];
             minIndex=i;
         }
    }
//    cout<<"minindex="<<minIndex<<endl;
    return minIndex; //return minimum index
}


void Graph::assignKeyValues()
{
    for(int i=0;i<rows;i++)
    {
        keys[i]=INT_MAX;
        VisitedArray[i]=false;
    }

}
void prims()
{
    //First assign key values
    graph1.assignKeyValues();

    //Second set the source key value to 0 and set parent of source node to -1
    graph1.keys[0]=0;
    graph1.parentArray[0]=-1;

    for(int i=0;i<graph1.rows;i++)
    {
        //find the minimum among all node
        int u = graph1.findMinimum();
        cout<<"u= "<<u<<endl;

        //set minimum node to visited node
        graph1.VisitedArray[u]=true;

        for(int v=0;v<graph1.rows;v++)
        {
            //cout<<"visited array = "<<graph1.VisitedArray[v]<<" AdjMat = "<<graph1.adjMatrix[u][v]<<" keys "<<graph1.keys[v]<<endl;
            if(graph1.VisitedArray[v]==false && graph1.adjMatrix[u][v]!=0 && graph1.adjMatrix[u][v]<graph1.keys[v])
            {
                graph1.parentArray[v]=u; //parent node of vth node will be u
                cout<<"u= "<<u<<" v= "<<v<<" parent[v]= "<<graph1.parentArray[v]<<endl;
                graph1.keys[v]=graph1.adjMatrix[u][v]; //update the value of vertex to minimum value
            }

        }
    }
}

void printMST()
{
    for(int i=0;i<graph1.rows;i++)
    {
        cout<<graph1.parentArray[i]<<" ";
    }
    cout<<endl;
}
int main()
{
    graph1.initializeMatrix();
    graph1.addEdge(0,1,7);
    graph1.addEdge(0,2,9);
    graph1.addEdge(0,5,14);
    graph1.addEdge(1,2,10);
    graph1.addEdge(1,3,15);
    graph1.addEdge(2,3,11);
    graph1.addEdge(2,5,2);
```

```
        graph1.addEdge(3,4,6);
        graph1.addEdge(4,5,9);

        graph1.printAdjMatrix();

        prims();
        printMST();
    }

```