Assignment No. 6

You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

```
*/
#include<stdlib.h>
#include<iostream>
#define inf 9999
using namespace std;

class prims
{
public:
            int cost[10][10],vertex,edge;
            void creategraph();
            void primfun(int);
};


void prims::creategraph()
{
            int v1,v2,i,j,wt;
            cout<<"\n\t\t Enter the MAXIMUM no of offices:::";
            cin>>vertex;
            cout<<"\n\t\t Enter the no of lease lines:::";
            cin>>edge;
   //       Create adjacency matrix with all values (except diagonals)=infinity
            for(i=0;i<vertex;i++)
                        for(j=0;j<vertex;j++)
                         {
                                    if(i==j)
                                                cost[i][j]=0;
                                                else
                                                 cost[i][j]=inf;
                                   }
            //enter information about edges and store cost of edges in adj matrix
            for(i=0;i<edge;i++)
            {
                        cout<<"\n\t\t Enter the lease lines (edge) and Their costs(v1,v2,wt)::";
                        cin>>v1>>v2>>wt;
                        cost[v1][v2]=cost[v2][v1]=wt;
            }
}

void prims::primfun(int s)
{
  int min,i,j,n=1,visited[10],dist[10],from[10],nextnode,mstcost=0;

//initialization of visited distance and from array
  for(i=0;i<vertex;i++)
  {
      visited[i]=0;
      dist[i]=inf;
      from[i]=s;
     }

     //starting  node visited
    visited[s]=1;

    //update distance array to reflect distance of any node from start node

            for(i=0;i<vertex;i++)
            {
                    if(visited[i]==0&&cost[s][i]<dist[i])
                    {
                            dist[i]=cost[s][i];

                    }


            }


      while(n<vertex) //To include vertex-1 no of edges in MST
      {
            min=inf;

            //find next node to visit (select node with minimum distance)
            for(i=0;i<vertex;i++)
            {
                    if(visited[i]==0&&dist[i]<min)
                    {
                            min=dist[i];
                            nextnode=i;

                    }
            }


            cout<<endl<<from[nextnode]<<"  "<<nextnode<<endl;//print edge included in MST
            n++;
            visited[nextnode]=1;
            mstcost+=dist[nextnode];                    //update MSTcost

   //update distance array with respect to new node which we have visited recently

            for(i=0;i<vertex;i++)
            {
                    if(visited[i]==0 && dist[i]>cost[nextnode][i])
                    {
                    dist[i]=cost[i][nextnode];
                    from[i]=nextnode;
                    }

            }


   }//while end
      cout<<"\n\tCost of minimun spanning tree is::"<<mstcost<<endl;

   }

   int main()
   {
      prims s1;

      int ch;
            while(1)
            {

                     cout<<"\n 1.Creategraph (Adjacency matrix form) \n 2.Prims Algorithm \n 3.Exit.\n";
                     cout<<"\n\nEnter Ur Choice= ";
                     cin>>ch;
                     switch(ch)
                     {

                     case 1:
                             s1.creategraph();
                                     break;

                     case 2:
                               int start;
                               cout<<"\nEnter Starting Vertex=";
                               cin>>start;
                               cout<<endl<<"MST is: "<<endl;
                               s1.primfun(start);
                                     break;

                     case 3:exit(0);
                     }
             }
      return 0;
   }
```

** **<u>Output:</u>** ***

```
/*
 1.Creategraph (Adjacency matrix form)
 2.Prims Algorithm
 3.Exit.


Enter Ur Choice= 1

           Enter the MAXIMUM no of offices:::4

           Enter the no of lease lines:::4

           Enter the lease lines (edge) and Their costs(v1,v2,wt)::0 1 14
```

```
                Enter the lease lines (edge) and Their costs(v1,v2,wt)::1 2 4

                Enter the lease lines (edge) and Their costs(v1,v2,wt)::2 3 34

                Enter the lease lines (edge) and Their costs(v1,v2,wt)::3 0 16

 1.Creategraph (Adjacency matrix form)
 2.Prims Algorithm
 3.Exit.


Enter Ur Choice= 2

Enter Starting Vertex=0

MST is:

0  1

1  2

0  3

        Cost of minimun spanning tree is::34

 1.Creategraph (Adjacency matrix form)
 2.Prims Algorithm
 3.Exit.


Enter Ur Choice= 3

*/
```