# PRIMS ALGORITHM

/*  **Name : Sagar Barapatre**
**Class : SE-10**
**Batch : E-10**
**Roll no. : 2007**
*/

**//GRAPH ADT**

```
/*
 * Graph.h
 *
 *  Created on: 20-Mar-2018
 *      Author: sagar
 */

#ifndef GRAPH_H_
#define GRAPH_H_

#include<iostream>
#include<cstring>

using namespace std;


typedef struct Friend
{
	string name;
	int dist;
	Friend * next;
	bool vis;

	struct Node * node_struct_pointer;

}Friend;

typedef struct Node
{
	string name;
	int id;
	Node * down;
	Friend * next;
	bool vis;
}Node;

Node* createnode(string,int);


Friend* createfriend(string);

class Graph {
	Node * root;
	int n;
	int cost;
public:
	Graph();
	Node* search(string);
	void accept();
	void addfriend(Node *);
```

```
        void display();
        virtual ~Graph();
        void prim();
        void display_mst();
};

#endif /* GRAPH_H_ */
```

```cpp
/*
 * Graph.cpp
 *
 *  Created on: 20-Mar-2018
 *      Author: sagar
 */

#include "Graph.h"
#include<iomanip>
Graph::Graph() {
	// TODO Auto-generated constructor stub
	root=NULL;
	n=0;
	cost=0;

}

Graph::~Graph() {
	// TODO Auto-generated destructor stub
}
Node* createnode(string tempname,int d)
{
	Node* newnode= new Node;
	newnode->next=NULL;
	newnode->down=NULL;
	newnode->name=tempname;
	newnode->id=d;
	newnode->vis=false;
	return newnode;
}

Node* Graph::search(string tempname)
{
	if(root->name==tempname)
		return root;

	Node* temproot=root;

	while(temproot!=NULL)
	{
		if(temproot->name==tempname)
		{
			return temproot;
		}
		temproot=temproot->down;
	}

	return NULL;
}


Friend* createFriend(string s,int d)
{
	Friend *newfriend=new Friend;
	newfriend->next=NULL;

	newfriend->name=s;
	newfriend->dist=d;
	newfriend->vis=false;
```

```
	return newfriend;
}
void Graph::accept()
{
	cout<<"Enter number of cities : ";
	cin>>n;
	int d=0;
	if(n>=1)
	{

		cout<<"\nEnter data of "<<1<<" th city ";

			string name;

			cout<<"\nEnter name of the city : ";
			cin>>name;

			root=createnode(name,d);
			d++;
	}

	Node* temproot=root;

	for(int i=1;i<n;i++)
	{
		cout<<"\nEnter data of "<<i+1<<" th city ";

		string name;

		cout<<"\nEnter name of the city : ";
		cin>>name;

		Node* newnode=createnode(name,d);
		d++;

			temproot->down=newnode;
			temproot=newnode;

	}

	temproot=root;

	for(int i=0;i<n;i++)
	{
		int f;
		cout<<"\nEnter cities to which "<<temproot->name<<" is connected :
";

		addfriend(temproot);
		temproot=temproot->down;
	}
}

void Graph::addfriend(Node *temp)
{
	cout<<"\nEnter name of the city or (-1) if there are no more directly
connected city : ";
	string fname;
	cin>>fname;
	if(fname=="-1")
		return;
```

```
	cout<<"\nEnter distance : ";
	int dist;
	cin>>dist;
	temp->next=createFriend(fname,dist);
	temp->next->node_struct_pointer=search(fname);

	Friend *tempfriend=temp->next;
	while(1)
	{
		cout<<"\nEnter name of the city or (-1) if there are no more
directly connected city : ";
		string fname;
		cin>>fname;
		if(fname=="-1")
			return;
		cout<<"\nEnter distance : ";
		int dist;
		cin>>dist;
		Friend *nextfriend=createFriend(fname,dist);
		nextfriend->node_struct_pointer=search(fname);
		tempfriend->next=nextfriend;
		tempfriend=nextfriend;
	}
}

void Graph :: display()
{
	Node *temproot;

	temproot=root;

	while(temproot != NULL)
	{
		cout<<temproot->name;

		Friend  *f;
			f=temproot->next;

			while(f != NULL)
			{
				cout<<"<- "<<f->name;
				f=f->next;
			}

		cout<<endl;
		temproot=temproot->down;
	}

	cout<<"NULL";
	cout<<endl;
}

void Graph :: prim()
{
	root->vis=1;	//first vertex of mst

	bool done=false;

	while(! done)
	{
		done=true;
```

```
		Node * current;
		current=root;
		Friend * min_edge=NULL;

		int min_dist=1000000;

		while(current != NULL)
		{
			if(current->vis == true)
			{

				Friend *f;
				f=current->next;

				while(f != NULL)
				{

					if((f->node_struct_pointer)->vis == false)
					{
						done=false;
						if(f->dist < min_dist)
						{
							min_dist=f->dist;
							min_edge=f;
						}
					}

					f=f->next;
				}
			}

			current = current->down;
		}

		if(min_edge != NULL)
		{
			min_edge->vis=1;
			cout<<"This is marked in friends : "<<min_edge-
>name<<endl<<endl;
			(min_edge->node_struct_pointer)->vis=1;
		}
	}

}


void Graph :: display_mst()
{
	Node *temproot;
		temproot=root;

		while(temproot != NULL)
		{
			cout<<setw(5)<<""<<temproot->name<<" :: ";

			Friend  *f;
				f=temproot->next;

				while(f != NULL )
				{
					if(f->vis==1)
					{
```

```
                                    cout<<setw(5)<<"<- "<<f->name;
                                    cout<<"( "<<f->dist<<" )"<<"";

                                    cost += f->dist;
                              }
                              f=f->next;
                        }

                  cout<<"\n";
                  temproot=temproot->down;
            }

            cout<<"Min dist : "<<cost<<endl;
            cout<<"\n";
}
```

**//MAIN FILE**

```
//============================================================================
// Name        : Prim.cpp
// Author      : sagar
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
#include<stdlib.h>
#include "Graph.h"
using namespace std;
int main()
{
      Graph g;
      do
            {
                  int ch;
                  cout<<"\tMENU\n";

                  cout<<"\t\t1.Enter Adjacency list\n\t\t2.Display
Graph\n\t\t3.Prim\n\t\t"
                              "4.Display minimum spanning tree"
                              "\n\t\t5.Re_Enter\n\t\t6.Exit\n";
                  cout<<"Enter Choice : ";
                  cin>>ch;
                  cout<<endl;

                  switch(ch)
                  {

                        case 1:
                                          g.accept();
                                          break;

                        case 2:
                                          g.display();
                                          break;

                        case 3:
                                          g.prim();
                                          break;

                        case 4:
                                          g.display_mst();
                                          break;

                        case 5:
                                          main();
                                          break;

                        case 6:
                                          exit(0);
                                          break;

                  }
            }
            while(1);
      return 0;
}
```