#include <iostream>

#include <cstring>

using namespace std;

/\*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.

\*/

class node

{

public:

node \*next;

int dist;

char name[10];

node(int x)

{

dist=x;

next=NULL;

}

friend class graph;

};

class graph

{

public:

node \*head[10];

int a[10][3],l,ch[15];

graph()

{

l=0;

for(int i=0;i<10;i++){

head[i]=NULL;

}

for(int i=0;i<15;i++)

{

ch[i]=-1;

}

}

int add(char so[],node \*t)

{

int j,i;

for( i=0;i<10;i++)

{

if(head[i]!=NULL)

{

j=strcmp(head[i]->name,so);

if(j==0)

{

node \*temp;

temp=head[i];

while(temp->next!=NULL)

temp=temp->next;

temp->next=t;

t->next=NULL;

return i;

}

}

else{

node \*tem;

tem=new node(9999);

strcpy(tem->name,so);

head[i]=tem;

head[i]->next=t;

return i;

}

}

return i;

}

void insert()

{

char so[10],de[10];

int amt,i,j;

cout<<"Enter Source city"<<endl;

cin>>so;

cout<<"Enter Destination City"<<endl;

cin>>de;

cout<<"Enter Money"<<endl;

cin>>amt;

node \*temp;

temp=new node(amt);

strcpy(temp->name,de);

i=add(so,temp);

node \*temp1;

temp1=new node(amt);

strcpy(temp1->name,so);

j=add(de,temp1);

a[l][0]=amt;

a[l][1]=i;

a[l][2]=j;

l++;

}

void display()

{

for(int i=0;i<4;i++)

{

cout<<a[i][0]<<" "<<a[i][1]<<" "<<a[i][2];

cout<<endl;

}

/\*int x;

for(int i=0;i<10;i++)

{

if(head[i]!=NULL)

{

node \*temp;

temp=head[i];

cout<<"\nSource: ";

cout<<temp->name<<endl;

temp=temp->next;

cout<<"Destinations: ";

while(temp!=NULL)

{

cout<<temp->name<<" "<<temp->dist<<" ";

temp=temp->next;

}

}

else{

break;

}

}

void display()

{

for(int i=0;i<20;i++)

{

node \*temp = head[i];

if(temp!=NULL)

{

cout<<"|"<<temp->name<<","<<temp->data<<"|"<<"->";

}

cout<<endl;

}

}

\*/

}

void swap(int &a,int &b){

int temp;

temp=a;

a=b;

b=temp;

}

void sort()

{

int x;

for(int i=1;i<l;i++){

for(int j=0;j<l-i;j++){

if(a[j+1][0]<a[j][0]){

swap(a[j][0],a[j+1][0]);

swap(a[j][1],a[j+1][1]);

swap(a[j][2],a[j+1][2]);

}

}

}

for(int i=0;i<l;i++){

x=a[i][1];

cout<<head[x]->name<<" ";

x=a[i][2];

cout<<head[x]->name<<" ";

cout<<a[i][0]<<endl;

}

}

void detect\_cycle(int z)

{

int p=a[z][1];

int r;

int q=a[z][2];

int s = parent(p);

int d = parent(q);

if(s==d)

{

return;

}

else

{

ch[s]=d;

cout<<head[p]->name<<" ";

cout<<head[q]->name<<" ";

r=a[z][0];

cout<<r<<endl;

}

}

void kruskal()

{

sort();

cout<<"\nMST:"<<endl;

for(int i=0;i<l;i++){

detect\_cycle(i);

}

}

int parent(int);

};

int graph::parent(int k)

{

if(ch[k]==-1)

{

return k;

}

else

{

return parent(ch[k]);

}

}

int main() {

graph g;

int ch;

int c=1;

while(c==1){

cout<<"1.Insert"<<endl;

cout<<"2.Display"<<endl;

cout<<"3.MST by Kruskal"<<endl;

cout<<"4.Exit"<<endl;

cin>>ch;

switch(ch)

{

case 1:{

g.insert();

break;

}

case 2:{

g.display();

break;

}

case 3:{

g.kruskal();

break;

}

case 4:{

c=0;

}

}

}

return 0;

}