#include <iostream>

#include<stdio.h>

using namespace std;

#include<stdlib.h>

#include<string.h>

#include<math.h>

#define MaxVnum 100

#define LEN sizeof(Node)

struct views{

int num;

char name[50];

char introduce[50];

int x;

int y;

};

typedef views dataType;

typedef struct {

dataType vex[100];

int edge[100][100];

int n,e;

}Graph;

Graph b;

int locatevex(Graph G,int v)

{

for(int i=0;i<G.n;i++)

{

if(v==G.vex[i].num)

{

return i;

}

}

return -1;

}

void Create(Graph &G)//创建无向图的邻接矩阵

{

int i,j,w;

int u,v;

cout << "请输入顶点数："<<endl;

cin>>G.n;

cout << "请输入边数:"<<endl;

cin>>G.e;

cout<<"请输入地点编号，地点名字，说明，X坐标，Y坐标，建立信息表：\n"<<endl;

for(i=0;i<G.n;i++)

{

cin>>G.vex[i].num>>G.vex[i].name>>G.vex[i].introduce>>G.vex[i].x>>G.vex[i].y;

}

cout << "请输入顶点信息:"<<endl;

for(int i=0;i<G.n;i++)

{

scanf("%d",&G.vex[i]);//输入顶点信息，存入顶点信息数组

}

for(int i=0;i<G.n;i++)//初始化邻接矩阵所有值为0，若是网，则初始化为无穷大

for(int j=0;j<G.n;j++)

if(i!=j)

G.edge[i][j]=1000;

else

G.edge[i][j]=0; //注意i==j时，设置为0

cout << "请输入每条边依附的两个顶点及权值："<<endl;

while(G.e--)

{

cin>>u>>v>>w;

i=locatevex(G,u);//查找顶点u的存储下标

j=locatevex(G,v);//查找顶点v的存储下标

if(i!=-1&&j!=-1)

G.edge[i][j]=G.edge[j][i]=w; //无向图邻接矩阵存储权值

}

}

void print(Graph G)

{

cout<<"邻接矩阵为:\n\n";

int i,j;

for(i=0;i<G.n;i++)

{

for(j=0;j<G.n;j++)

{

cout<<G.edge[i][j]<<" ";

if(j+1==G.n)

{

printf("\n");

}

}

}

cout<<"\n\n"<<endl;

}

int dist[MaxVnum][MaxVnum],p[MaxVnum][MaxVnum];

void Floyd(Graph G) //用Floyd算法求无向网G中各对顶点i和j之间的最短路径

{

int i,j,k;

for(i=0;i<G.n;i++) //各对结点之间初始已知路径及距离

for(j=0;j<G.n;j++)

{

dist[i][j]=G.edge[i][j];

if(dist[i][j]<1000 && i!=j)

p[i][j]=i; //如果i和j之间有弧，则将j的前驱置为i

else p[i][j]=-1; //如果i和j之间无弧，则将j的前驱置为-1

}

for(k=0;k<G.n; k++)

for(i=0;i<G.n; i++)

for(j=0;j<G.n; j++)

if(dist[i][k]+dist[k][j]<dist[i][j])//从i经k到j的一条路径更短

{

dist[i][j]=dist[i][k]+dist[k][j]; //更新dist[i][j]

p[i][j]=p[k][j]; //更改j的前驱为k

}

}

typedef struct Node

{

char elem[20];

double distance;

struct Node \*next;

int x,y;

};

void Sort(Node \*head)

{

Node \*p,\*q;

int t,m,n;

char str[20];

if(head==NULL)

{

return;

}

for(p=head;p!=NULL;p=p->next)

{

for(q=head;q->next!=NULL;q=q->next)

{

if(p->distance<q->distance)

{

t=p->distance;

p->distance=q->distance;

q->distance=t;

strcpy(str,p->elem);

strcpy(p->elem,q->elem);

strcpy(q->elem,str);

m=p->x;

p->x=q->x;

q->x=m;

n=p->y;

p->y=q->y;

q->y=n;

}

}

}

}

int k;

Node \* Search(Graph G)

{

int x,y;

cout<<"请输入用户的地理位置"<<endl;

scanf("%d %d",&x,&y);

cout<<"请输入网点类型"<<endl;

char str[20];

cin>>str;

cout<<"请输入显示网点的个数"<<endl;

int cnt=0;

cin>>k;

int b[20]={0};

Node \*head=NULL,\*p1=NULL,\*p2=NULL;

for(int i=0;i<G.n;i++)

{

if(strcmp(str,G.vex[i].introduce)==0)

{

cnt++;

int x1=x-G.vex[i].x;

int m=x1\*x1;

int y1=y-G.vex[i].y;

int n=y1\*y1;

double sum=m+n;

double num;

num=sqrt(sum);

p1=(Node \*)malloc(LEN);

p1->next=NULL;

strcpy(p1->elem,G.vex[i].name);

p1->distance=num;

p1->x=G.vex[i].x;

p1->y=G.vex[i].y;

if(cnt==1)

{

head=p1;

}

else

{

p2->next=p1;

}

p2=p1;

}

}

if(k>cnt)

{

printf("输入有误！");

}

return head;

}

void Printf(Node \* head,int k)

{

if(head==NULL)

{

return;

}

struct Node \*p=head;

while(k--)

{

printf("%s:%.0lf ",p->elem,p->distance);

cout<<"坐标为: ("<<p->x<<","<<p->y<<")\n";

p=p->next;

}

}

void DisplayPath(Graph G,int s,int t )//显示最短路径

{

if(p[s][t]!=-1)

{

DisplayPath(G,s,p[s][t]);

printf("%d",G.vex[p[s][t]]);

printf("-->");

}

}

void show(Graph G)

{

Floyd(G);

int num1,num2;

cout<<"请输入用户所在的位置的编号\n";

cin>>num1;

cout<<"请输入选择的网点的编号\n";

cin>>num2;

int x=locatevex(G,num1);

int y=locatevex(G,num2);

DisplayPath(G,x,y);

printf("%d",G.vex[y]);

cout<<"最短路径长度为："<<dist[x][y]<<endl;

cout<<endl;

}

int main()

{

Node \*head;

Create(b);

print(b);

head=Search(b);

Sort(head);

Printf(head,k);

show(b);

return 0;

}