**國立清華大學**

**電機工程學系**

**資料結構Homework1**

**學號：103061223**

**姓名：李俊穎(Lee Junying)**

**All of my source codes：**

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <fstream>

#include <algorithm>

#include <climits>

using namespace std;

int D[1000][1000];

int find\_k\_ary\_neighbors(int from , int hopes);

int find\_d\_radius\_neighbors(int from , int distance);

int total\_neighbors;

int visited[1000];

int NUM\_CITY = 100;

int MAX\_HOPS = 3;

int MAX\_DISTANCE = 100;

int main()

{

ifstream in\_file ("highway\_map");

string line;

int first, second, third,total=0;

char c\_line[1000];

if (!in\_file.is\_open()) exit(-1);

for(int m=1; m<=NUM\_CITY; m++){

for(int n=1; n<=NUM\_CITY; n++){

D[m][n] = INT\_MAX;

}

}

int total\_link = 0;

while ( getline (in\_file,line) ){

// cout << line << '\n';

strcpy(c\_line, line.c\_str());

sscanf(c\_line, "(%d,%d,%d)", &first, &second, &third);

printf("(%d, %d, %d)\n", first, second, third);

D[first][second] = third;

D[second][first] = third;

total\_link++;

}

// Copy D[m][n] to D[n][m]

for(int m=1; m<=NUM\_CITY; m++){

for(int n=1; n<=NUM\_CITY; n++){

if(m!=n && m < n && D[m][n] != INT\_MAX)

D[n][m] = D[m][n];

}

}

cout<<"for 3-ary neighbors:"<<endl;

// find\_out\_k\_ary\_neighbors

for(int from=1; from<=NUM\_CITY; from++){

find\_k\_ary\_neighbors(from, MAX\_HOPS);

if(from==3||from==29||from==75)

{

cout<<"C"<<from<<" is linked to :";

for(int a=1;a<=NUM\_CITY;a++)

if(visited[a]==1&&a!=3&&a!=29&&a!=75){

cout<<a<<" ";

total++;

}

cout<<endl<<"total neighbors:"<<total<<endl<<endl;

total=0;

}

for(int a=1;a<=1000;a++)

{

visited[a]=0;

}

}

cout<<"for 100 radius neighbors:"<<endl;

// find\_out\_d\_radius\_neighbors

for(int from=1; from<=NUM\_CITY; from++){

find\_d\_radius\_neighbors(from, MAX\_DISTANCE);

if(from==3||from==29||from==75)

{

cout<<"C"<<from<<" is linked to :";

for(int a=1;a<=NUM\_CITY;a++)

if(visited[a]==1&&a!=3&&a!=29&&a!=75){

cout<<a<<" ";

total++;

}

cout<<endl<<"total neighbors:"<<total<<endl<<endl;

total=0;

}

for(int a=1;a<1000;a++)

{

visited[a]=0;

}

}

return 0;

}

int find\_k\_ary\_neighbors(int from,int hopes)

{

visited[from]=1;

for(int m=1; m<=NUM\_CITY; m++){

if(D[from][m]!=INT\_MAX)

{

visited[m]=1;

hopes--;

if(hopes>0){

find\_k\_ary\_neighbors(m,hopes);

hopes++;

}

}

}

}

int find\_d\_radius\_neighbors(int from ,int distance)

{

int distance1=0;

visited[from]=1;

for(int m=1;m<=NUM\_CITY;m++)

{

if(D[from][m]!=INT\_MAX)

{

if(visited[m]==1)

{

continue;

}

else

{

distance1=distance-D[from][m];

}

if(distance1>0)

{

visited[m]=1;

//cout<<distance<<endl;

find\_d\_radius\_neighbors(m,distance1);

distance1=distance1+D[from][m];

}

else

{

distance1=distance1+D[from][m];

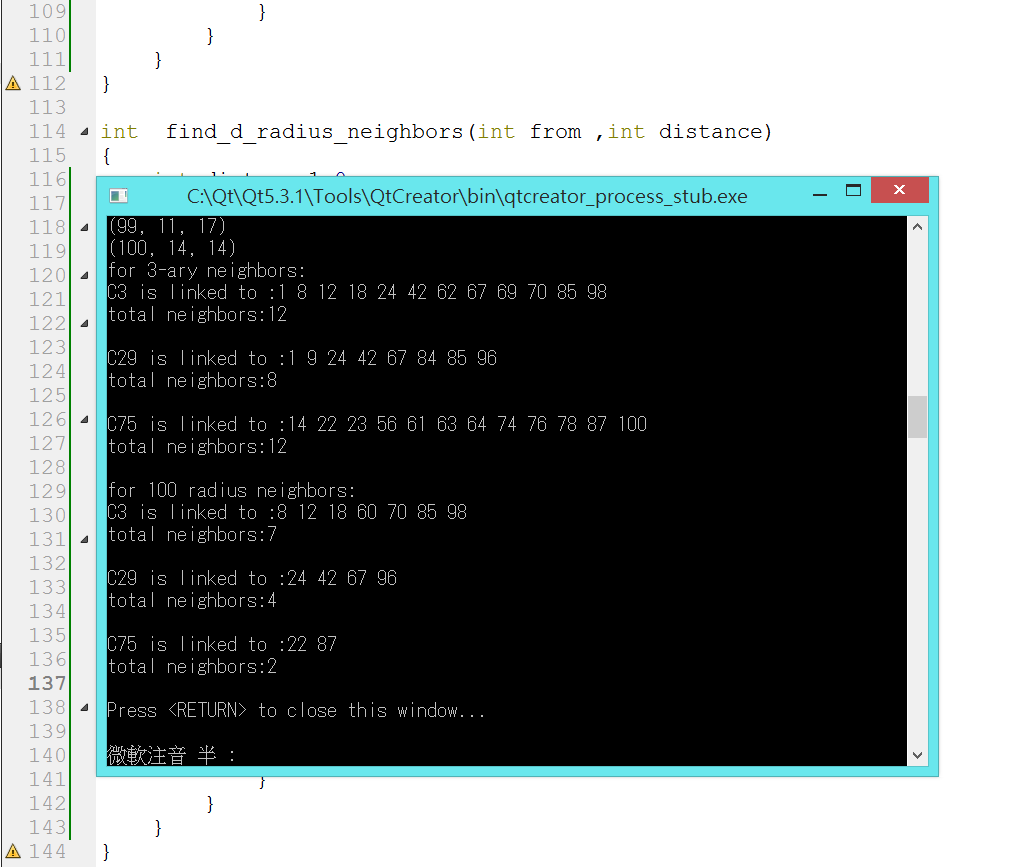
}

}

}

}

**The execution results of my program：**

****

**想法：**

第一小題是要找出與C3，C29，C75相連三步以內的城市。而一開始用for迴圈先呼叫function，並且傳入一開始要找的城市以及他的階層數MAX\_HOPES，並且將一開始的起始城市visited設為1，以防下次進入重複的尋找路徑。接下來，function內再用for迴圈將全部的城市都掃過一次，如果他們之間的距離D不是無限大(INT\_MAX)，則代表這兩個城市是有相連的，於是將其設置為拜訪過的鄰居(將其visited存入1)，並將階層數hopes減1。如果階層數沒有被減到0，代表尚未找完三步以內的鄰居，所以再設一個判斷條件並再次呼叫自己一次(recursive)，並在底下將hopes加回1，如此才能在搜索到底部時，回到上一層繼續往另外一條路找。之後再將全部visited陣列裡數字是1的全部印出來，即為解答。

第2題的做法與第一題就大同小異了，幾乎完全一樣，唯一不同的是為了防止重複無限迴圈，我設置如果找到原本visited陣列裡已經是1的城市，就讓其跳過那次的for迴圈，且以最大距離100往下減去做判斷，如果大於0，代表就是在100公里以內的鄰居。這次的想法大概如上，謝謝。