**Einbahnstrasse**

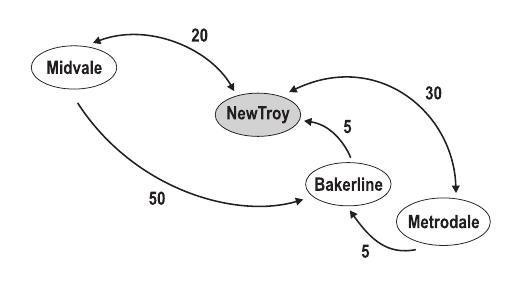
**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 3079    Accepted Submission(s): 960**

Problem Description

Einbahnstra http://acm.hdu.edu.cn/data/images/con208-1006-1.JPGe (German for a one-way street) is a street on which vehicles should only move in one direction. One reason for having one-way streets is to facilitate a smoother flow of traffic through crowded areas. This is useful in city centers, especially old cities like Cairo and Damascus. Careful planning guarantees that you can get to any location starting from any point. Nevertheless, drivers must carefully plan their route in order to avoid prolonging their trip due to one-way streets. Experienced drivers know that there are multiple paths to travel between any two locations. Not only that, there might be multiple roads between the same two locations. Knowing the shortest way between any two locations is a must! This is even more important when driving vehicles that are hard to maneuver (garbage trucks, towing trucks, etc.)  
  
You just started a new job at a car-towing company. The company has a number of towing trucks parked at the company's garage. A tow-truck lifts the front or back wheels of a broken car in order to pull it straight back to the company's garage. You receive calls from various parts of the city about broken cars that need to be towed. The cars have to be towed in the same order as you receive the calls. Your job is to advise the tow-truck drivers regarding the shortest way in order to collect all broken cars back in to the company's garage. At the end of the day, you have to report to the management the total distance traveled by the trucks.

Input

Your program will be tested on one or more test cases. The first line of each test case specifies three numbers (N , C , and R ) separated by one or more spaces. The city has N locations with distinct names, including the company's garage. C is the number of broken cars. R is the number of roads in the city. Note that 0 < N < 100 , 0<=C < 1000 , and R < 10000 . The second line is made of C + 1 words, the first being the location of the company's garage, and the rest being the locations of the broken cars. A location is a word made of 10 letters or less. Letter case is significant. After the second line, there will be exactly R lines, each describing a road. A road is described using one of these three formats:  
  
  
A -v -> B  
A <-v - B  
A <-v -> B  
  
  
A and B are names of two different locations, while v is a positive integer (not exceeding 1000) denoting the length of the road. The first format specifies a one-way street from location A to B , the second specifies a one-way street from B to A , while the last specifies a two-way street between them. A , ``the arrow", and B are separated by one or more spaces. The end of the test cases is specified with a line having three zeros (for N , C , and R .)  
  
The test case in the example below is the same as the one in the figure.



Output

For each test case, print the total distance traveled using the following format:  
  
  
k . V  
  
  
Where k is test case number (starting at 1,) is a space, and V is the result.

Sample Input

4 2 5

NewTroy Midvale Metrodale

NewTroy <-20-> Midvale

Midvale --50-> Bakerline

NewTroy <-5-- Bakerline

Metrodale <-30-> NewTroy

Metrodale --5-> Bakerline

0 0 0

Sample Output

1. 80

Source

[2008 ANARC](http://acm.hdu.edu.cn/search.php?field=problem&key=2008+ANARC&source=1&searchmode=source)

分析：从起点到某地，然后在倒过来 求最短路的和。

重点是一个地方可能去多次，所以要对要去的地方进行遍历。

同样的用map容器 构造一个有向图 就清楚了

AC代码：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <map>

#define INF 0x3f3f3f3f

#define MAX 1000

using namespace std;

int vis[MAX],dis[MAX];

int map1[MAX][MAX];

char rr[1001][100];

int n,m,C;

void floyd()

{

for(int k = 1; k <= n; k++)

for(int i = 1; i <= n; i++)

for(int j = 1; j <= n; j++)

if(map1[i][j] > map1[i][k]+map1[k][j])

map1[i][j] = map1[i][k]+map1[k][j];

}

int main()

{

int case1 = 1;

while(~scanf("%d%d%d",&n,&C,&m))

{

if(n==0&&C==0&&m==0)

break;

map<string,int>s;

int ans = 1;

s.clear();

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

{

scanf("%s",rr[i]);

if(rr[i]==0)

s[rr[i]] = ans++;

}

for(int i = 0; i < 110; i++)

{

for(int j = 0; j < 110; j++)

map1[i][j] = INF;

map1[i][i] = 0;

}

char p[31],q[31];

char a[31];

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

{

scanf("%s%s%s",p,a,q);

if(!s[p]) s[p] = ans++;

if(!s[q]) s[q] = ans++;

int len = strlen(a);

int value = 0,flag = 1;

for(int j = len - 3; j > 1; j--)

{

value+=(a[j] - 48)\*flag;

flag\*=10;

}

if(map1[s[p]][s[q]] > value&&a[0] == '<')

map1[s[p]][s[q]] = value;

if(map1[s[q]][s[p]] > value&&a[len-1] == '>')

map1[s[q]][s[p]] = value;

}

/\*\*for(int i = 0; i < n; i++)

{

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

printf("%d ",map1[i][j]);

printf("\n");

}\*/

printf("%d. ",case1++);

floyd();

int sum = 0;

for(int i = 1; i <= C; i++)

{

sum+=map1[s[rr[0]]][s[rr[i]]] + map1[s[rr[i]]][s[rr[0]]];

}

printf("%d\n",sum);

}

return 0;

}