Problem D

Meeting Prof. Miguel

Input: standard input
Output: standard output
Time Limit: 5 seconds

I have always thought that someday I will meet Professor Miguel, who has allowed me to arrange so many contests. But I have managed to miss all the opportunities in reality. At last with the help of a magician I have managed to meet him in the magical **City of Hope**. The city of hope has many roads. Some of them are bi-directional and others are unidirectional. Another important property of these streets are that some of the streets are for people whose age is less than thirty and rest are for the others. This is to give the minors freedom in their activities. Each street has a certain length. Given the description of such a city and our initial positions, you will have to find the most suitable place where we can meet. The most suitable place is the place where our combined effort of reaching is minimum. You can assume that I am 25 years old and Prof. Miguel is 40+.

First meeting after five years of collaboration (Shanghai, 2005)

Input

The input contains several descriptions of cities. Each description of city is started by a integer N, which indicates how many streets are there. The next N lines contain the description of N streets.

The description of each street consists of four uppercase alphabets and an integer. The first alphabet is either 'Y' (indicates that the street is for young) or 'M' (indicates that the road is for people aged 30 or more) and the second character is either 'U' (indicates that the street is unidirectional) or 'B' (indicates that the street is bi-directional). The third and fourth characters, X and Y can be any uppercase alphabet and they indicate that place named X and Y of the city are connected (in case of unidirectional it means that there is a one-way street from X to Y) and the last non-negative integer C indicates the energy required to walk through the street. If we are in the same place we can meet each other in zero cost anyhow. Every energy value is less than 500.

After the description of the city the last line of each input contains two place names, which are the initial position of me and Prof. Miguel respectively.

A value zero for N indicates end of input.

Output

For each set of input, print the minimum energy cost and the place, which is most suitable for us to meet. If there is more than one place to meet print all of them in lexicographical order in the same line, separated by a single space. If there is no such places where we can meet then print the line “You will never meet.”

Sample Input:

```
4
Y U A B 4
Y U C A 1
M U D B 6
M B C D 2
A D
2
Y U A B 10
M U C D 20
A D
0
```

Sample Output

```
10 B
You will never meet.
```

Shahriar Manzoor

“All primes are odd except 2, which is the oddest of all.”

Rare Order

A rare book collector recently discovered a book written in an unfamiliar language that used the same characters as the English language. The book contained a short index, but the ordering of the items in the index was different from what one would expect if the characters were ordered the same way as in the English alphabet. The collector tried to use the index to determine the ordering of characters (i.e., the collating sequence) of the strange alphabet, then gave up with frustration at the tedium of the task.

You are to write a program to complete the collector's work. In particular, your program will take a set of strings that has been sorted according to a particular collating sequence and determine what that sequence is.

Input

The input consists of an ordered list of strings of uppercase letters, one string per line. Each string contains at most 20 characters. The end of the list is signalled by a line with the single character `#'. Not all letters are necessarily used, but the list will imply a complete ordering among those letters that are used.

Output

Your output should be a single line containing uppercase letters in the order that specifies the collating sequence used to produce the input data file.

Sample Input

```
XWY
ZX
ZXY
ZXW
YWWX
#
```

Sample Output

```
XZYW
```

Problem A: Freckles

In an episode of the Dick Van Dyke show, little Richie connects the freckles on his Dad's back to form a picture of the Liberty Bell. Alas, one of the freckles turns out to be a scar, so his Ripley's engagement falls through.

Consider Dick's back to be a plane with freckles at various (x,y) locations. Your job is to tell Richie how to connect the dots so as to minimize the amount of ink used. Richie connects the dots by drawing straight lines between pairs, possibly lifting the pen between lines. When Richie is done there must be a sequence of connected lines from any freckle to any other freckle.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line contains $0 < n <= 100$, the number of freckles on Dick's back. For each freckle, a line follows; each following line contains two real numbers indicating the (x,y) coordinates of the freckle.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Your program prints a single real number to two decimal places: the minimum total length of ink lines that can connect all the freckles.

Sample Input

```
1

3
1.0 1.0
2.0 2.0
2.0 4.0
```

Sample Output

```
3.41
```

The path in the colored field

The Problem

The square field consists of M×M cells. Each cell is colored in one of three colors (1,2,3). The initial state is chosen in one of the cells of color 1. In each step one allowed to move one cell up, down, left or right remaining inside the field.

You are to define the minimal amount of steps one should make to get a cell of color 3 independent on the initial state.

Note that the field contains at least one cell of color 1 and at least one cell of color 3.

The Input

The input consists of several input blocks. The first line of each block contains integer M, the size of the field. Then there are M lines with colors of the cells.

The Output

For each input block the output should consist of one line with the integer, the minimal amount of steps one should make to get a cell of color 3 independent on the initial state.

Sample Input

```
4
1223
2123
2213
3212
2
12
33
```

Sample Output

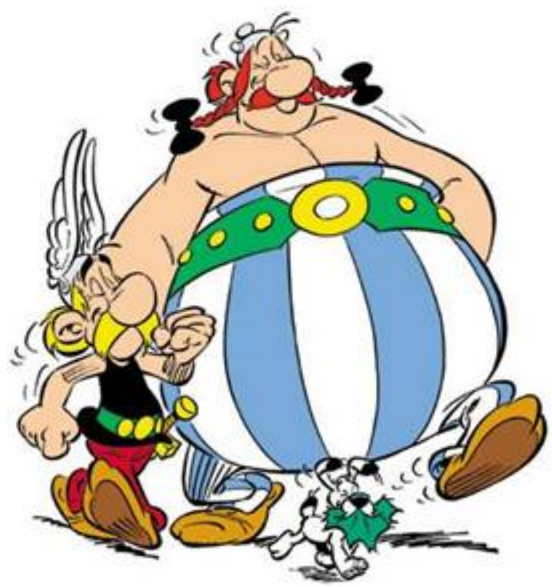
```
3
1
```

Problem A

Asterix and Obelix

Input: standard input
Output: standard output
Time Limit: 5 seconds
Memory Limit: 32 MB

After winning a gruesome battle against the Romans in a far-away land, Asterix and his dearest friend Obelix are now returning home. However Obelix is not with Asterix now. He has left Asterix in order to deliver menhir to one of his international buyers (as you probably know, recently he has extended his trade to international markets). But he has promised to join Asterix on his way home and Asterix has promised to host a feast for Obelix (you know how fat he is!) in the city they meet. Obelix may meet Asterix in any city on his way home including the starting and the destination city.



Now Asterix is sitting with a map and trying to figure out the cheapest route home. The map shows the cities and the cost (in sestertii) of going from one city to another if there is a road connecting them directly. For each city in the map Asterix has also calculated the cost (in sestertii) of hosting a feast for Obelix in that city. There will be only one feast and for safety Asterix has decided to set aside enough sestertii to host a feast in the costliest city on the route.

Since Asterix does not have a computer, he seeks your help to find out the cheapest route home.

Input

The input may contain multiple test cases.

The first line of each test case contains three integers **C (£ 80)**, **R (£ 1000)** and **Q (£ 6320)** where **C** indicates the number of cities (cities are numbered using distinct integers ranging from 1 to **C**), **R** represents the number of roads and **Q** is the number of queries.

The next line contains **C** integers where the **i**-th integer $\mathbf{f_i}$ is the cost (in sestertii) of hosting a feast in city **i**.

Each of the next **R** lines contains three integers: $\mathbf{c_1}$**,** $\mathbf{c_2}$ **(**$^{\mathbf{1}}$ $\mathbf{c_1}$**)** and **d** indicating that the cost of going from city $\mathbf{c_1}$ to $\mathbf{c_2}$ (or from $\mathbf{c_2}$ to $\mathbf{c_1}$) is **d** sestertii.

Each of the next **Q** lines contains two integers $\mathbf{c_1}$ and $\mathbf{c_2}$ **(**$\mathbf{c_1}$ $^{\mathbf{1}}$ $\mathbf{c_2}$**)** asking for the cost (in sestertii) of the cheapest route from city $\mathbf{c_1}$ to city $\mathbf{c_2}$.

The input will terminate with three zeros form **C**, **S** and **Q**.

Output

For each test case in the input first output the test case number (starting from 1) as shown in the sample output. Then for each query in the input print a line giving the minimum cost (in sestertii) of going from the first to the second city in the query. If there exists no path between them just print "`-1`".

Print a blank line between two consecutive test cases.

Sample Input

```
7 8 5
2 3 5 15 4 4 6
1 2 20
1 4 20
1 5 50
2 3 10
3 4 10
3 5 10
4 5 15
6 7 10
1 5
1 6
5 1
3 1
6 7
4 4 2
2 1 8 3
1 2 7
1 3 5
2 4 8
3 4 6
1 4
2 3
0 0 0
```

Sample Output

```
Case #1
45
```

```
-1
45
35
16

Case #2
18
20
```

(World Finals Warm-up Contest, Problem Setter: Rezaul Alam Chowdhury)

Critical Links

In a computer network a link L, which interconnects two servers, is considered critical if there are at least two servers A and B such that all network interconnection paths between A and B pass through L. Removing a critical link generates two disjoint sub-networks such that any two servers of a sub-network are interconnected. For example, the network shown in figure 1 has three critical links that are marked bold: `0 -1`, `3 - 4` and `6 - 7`.

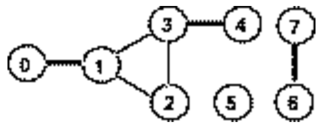


Figure 1: Critical links

It is known that:

1. the connection links are bi-directional;
2. a server is not directly connected to itself;
3. two servers are interconnected if they are directly connected or if they are interconnected with the same server;
4. the network can have stand-alone sub-networks.

Write a program that finds all critical links of a given computer network.

Input

The program reads sets of data from a text file. Each data set specifies the structure of a network and has the format:

$no\_of\_servers$

$server_0$ $(no\_of\_direct\_connections)$ $connected\_server$ $\ldots$ $connected\_server$

...

$server_{no\_of\_servers}$ $(no\_of\_direct\_connections)$ $connected\_server$ $\ldots$ $connected\_server$

The first line contains a positive integer $no\_of\_servers$ (possibly 0) which is the number of network servers. The next $no\_of\_servers$ lines, one for each server in the network, are randomly ordered and show the way servers are connected. The line corresponding to $server_k$, $0 \leq k \leq no\_of\_servers - 1$, specifies the number of direct connections of $server_k$ and the servers which are directly connected to $server_k$. Servers are represented by integers from 0 to $no\_of\_servers - 1$. Input data are correct. The first data set from sample input below corresponds to the network in figure 1, while the second data set specifies an empty network.

Output

The result of the program is on standard output. For each data set the program prints the number of critical links and the critical links, one link per line, starting from the beginning of the line, as shown in the sample output below. The links are listed in ascending order according to their first element. The output for the data set is followed by an empty line.

Sample Input

```
8
0 (1) 1
1 (3) 2 0 3
2 (2) 1 3
3 (3) 1 2 4
4 (1) 3
7 (1) 6
6 (1) 7
5 (0)

0
```

Sample Output

```
3 critical links
0 - 1
3 - 4
6 - 7

0 critical links
```

Miguel Revilla
2001-01-05

Tree Recovery

Little Valentine liked playing with binary trees very much. Her favorite game was constructing randomly looking binary trees with capital letters in the nodes.

This is an example of one of her creations:

```
                                               D
                                              / \
                                             /   \
                                            B     E
                                           / \     \
                                          /   \     \
                                         A     C     G
                                                    /
                                                   /
                                                  F
```

To record her trees for future generations, she wrote down two strings for each tree: a preorder traversal (root, left subtree, right subtree) and an inorder traversal (left subtree, root, right subtree).

For the tree drawn above the preorder traversal is `DBACEGF` and the inorder traversal is `ABCDEFG`.

She thought that such a pair of strings would give enough information to reconstruct the tree later (but she never tried it).

Now, years later, looking again at the strings, she realized that reconstructing the trees was indeed possible, but only because she never had used the same letter twice in the same tree.

However, doing the reconstruction by hand, soon turned out to be tedious.

So now she asks you to write a program that does the job for her!

Input Specification

The input file will contain one or more test cases. Each test case consists of one line containing two strings preord and inord, representing the preorder traversal and inorder traversal of a binary tree. Both strings consist of unique capital letters. (Thus they are not longer than 26 characters.)

Input is terminated by end of file.

Output Specification

For each test case, recover Valentine's binary tree and print one line containing the tree's postorder traversal (left subtree, right subtree, root).

Sample Input

```
DBACEGF ABCDEFG
BCAD CBAD
```

Sample Output

```
ACBFGED
CDAB
```

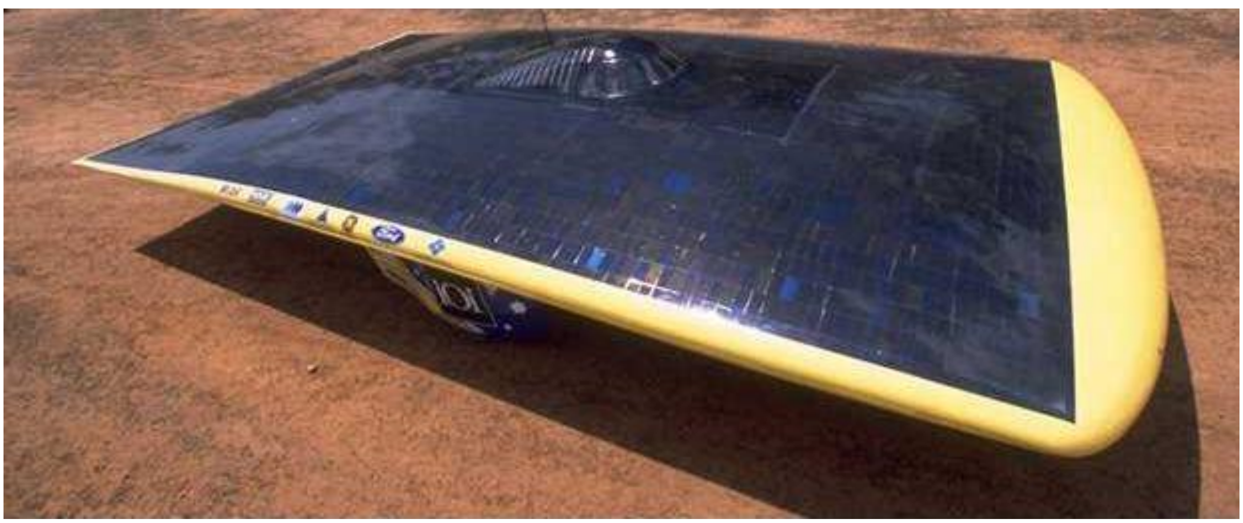
Problem E

Magic Car

Input: standard input
Output: standard output
Time Limit: 5 seconds

ACM (Association of Car Modernization) has recently developed a new car, "MAGIC CAR". It uses solar energy. The car has some interesting characteristics :

- It uses up constant amount of energy at the start for any initial speed.
- It also uses up constant energy when stops.
- If it changes speed to a value which is less than already achieved least speed or greater than already achieved most speed, it uses up some energy. In both cases the energy is equal to the absolute difference of the current speed and previously achieved least or most speed.
- The loss of energy does not depend on the distance the car covered (Really magic!!!).



Mr. Oberoy has such a magic car. So far he has used the car quite intelligently so that minimum energy is used up. This was easy for him as he could take any speed on any road. But recently, TCD(Transport Control Department) has decided that there will be a fixed speed for each road in the city and everybody must maintain the speed. Mr. Oberoy is in problem now. Can you help him so that he can optimally use the car ?

Input

Each dataset starts with two positive integer, **N (2<=N<=200)** denoting the number of junctions and **M (1<=M<=1000)** denoting the number of roads in Mr. Oberoy's city. Each junction is identified by a unique integer from 1 to N. In next few lines there will be road descriptions. A road is described by three positive integers which are start, end junctions and the fixed speed of that road. There may be more than one roads between two junctions. Roads are bidirectional. In the next line there will be two positive integers which are the used up energy during start and stop of the magic car. Next line will contain an integer **K (1<=K<=5)** indicating the number of queries. Each of following K lines will contain two integers, the source and destination junction of Mr. Oberoy. Source and destination will not be same. Input is terminated by EOF.

Output

For each dataset print the minimum possible used up energy for each query of Mr. Oberoy. It is guranteed that the destination is always reachable from source.

Sample Input

```
4 4
1 2 2
2 3 4
1 4 1
3 4 2
5 5
2
1 3
1 2
```

Sample Output

```
11
10
```

Author : Md. Kamruzzaman

The Real Programmers' Contest-2

Problem E
Sending email
Time Limit: 3 seconds

"A new internet watchdog is creating a stir in Springfield. Mr. X, if that is his real name, has come up with a sensational scoop."

`Kent Brockman`

There are **n** SMTP servers connected by network cables. Each of the **m** cables connects two computers and has a certain latency measured in milliseconds required to send an email message. What is the shortest time required to send a message from server **S** to server **T** along a sequence of cables? Assume that there is no delay incurred at any of the servers.

Input
The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n** (2<=**n**<20000), **m** (0<=**m**<50000), **S** (0<=**S**<**n**) and **T** (0<=**T**<**n**). **S**!=**T**. The next **m** lines will each contain 3 integers: 2 different servers (in the range [0, **n**-1]) that are connected by a bidirectional cable and the latency, **w**, along this cable (0<=**w**<=10000).

Output
For each test case, output the line "Case #**x**:" followed by the number of milliseconds required to send a message from **S** to **T**. Print "unreachable" if there is no route from **S** to **T**.

<table>
<tr><th>Sample Input</th><th>Sample Output</th></tr>
<tr><td><pre>3
2 1 0 1
0 1 100
3 3 2 0
0 1 100
0 2 200
1 2 50
2 0 0 1</pre></td><td><pre>Case #1: 100
Case #2: 150
Case #3: unreachable</pre></td></tr>
</table>

Problemsetter: Igor Naverniouk