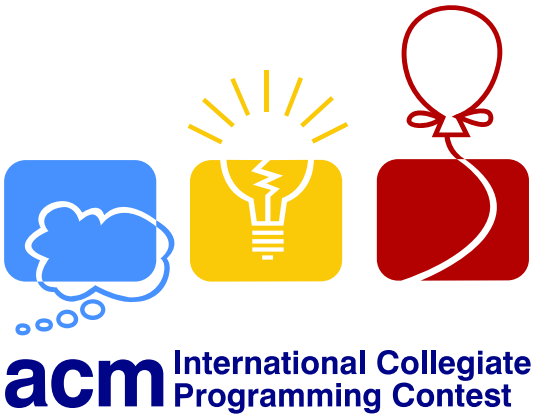
German Collegiate Programming Contest

2. Juli 2011



The Problem Set

Good luck and have fun!

This page is intentionally left (almost) blank.

Problem A

Faculty Dividing Powers

Fred Faculty and Paul Power love big numbers. Day after day Fred chooses a random integer n and he computes $n!$. His friend Paul amuses himself by computing several powers of his randomly chosen integer k like k^2, k^3, ... and so on. On a hot summer day, Fred and Paul got really, really bored, so they decided to play a joke on their buddy Dave Divider. Fred chooses a random integer n while Paul chooses a random integer k. They want Dave to find the biggest integer i such that k^i divides $n!$ without a remainder, otherwise they will throw a cake in Dave's face. Because Dave does not like cakes in his face, he wants you to help him finding that integer i.

Input

The first line contains the number of test cases t ($1 \leq t \leq 100$). Each of the following t lines contains the two numbers n, k ($2 \leq n \leq 10^{18}, 2 \leq k \leq 10^{12}$) separated by one space.

Output

For each test case, print the maximum integer i on a separate line.

Sample Input	Sample Output
2 5 2 10 10	3 2

This page is intentionally left (almost) blank.

Problem B

Genetic Fraud

Computer Scientists have a rough life. Due to the shortage of bright young minds like yourself, good programmers are mostly like famous movie stars. It's just like everyone wants to get their hands on your hard earned cash. The situation got so out of hand that you are faced by a lawsuit nearly every day of the year. Each lawsuit is filed by someone claiming alimonies for your alleged child. Luckily for you, you know pretty much about genetic sequences. You know that the human DNA sequence can be represented by a possibly $1 \leq N \leq 1\,000$ characters long string, containing only characters 'a' to 'z'. And you know that a similarity test of the DNA strings of the alleged child to your own can prove your innocence.

The only problem is that this does not only happen to you. As all labs are busy, testing needs at least a year. Still, not all hope is lost. You managed to get the information from one of the DNA labs on how to compute the probability of a genetic relationship between two DNA strings. If you could only help the DNA labs to test two DNA strings for a genetic relationship really fast, you could get the evidence you need for your own lawsuits.

A genetic relationship test, or GRT, requires some heavy computation on the DNA strings. They first acquire all similar regions within two DNA strings. We understand a region within a DNA string to be a consecutive interval within the DNA string. Regions of equal length of DNA strings are similar to each other, whenever $|(a[i] - b[j])| \leq 1$, where $a[i], b[j]$ are lower-case letters. A GRT between two DNA sequences is positive, whenever the two DNA sequences have a similar region of at least one half the length of the DNA sequences.

Input

The first line of the input gives the number of test cases C ($0 < C \leq 1\,000$). The first line of each such test case holds an integer N, denoting the length of the two DNA strings to be tested. The following two lines contain a string of N lower-case letters each, giving the two DNA substrings of the two persons to test.

Output

For each test case print one line. If the GRT is positive, print out "`POSITIVE`". Print "`NEGATIVE`", if the test fails.

Sample Input	Sample Output
3 4 aaaa bbcc 8 iacdefgh abeaaaaa 8 iacdefgh abeafaaa	POSITIVE NEGATIVE NEGATIVE

This page is intentionally left (almost) blank.

Problem C

Indiana Jones and the lost Soccer Cup

In 1930 the first FIFA World Cup was held in Uruguay, and Uruguay won the Cup in a dramatic final against Argentina. All of this seems like ages past, and now myth and legend rank around that historic first world cup. Some even claim that the original trophy cup that was awarded to Uruguay for their victory has mythical powers and would grant any side the strength to win the World Cup. Now that original cup has been lost in time, and even though its mythical powers are probably just stories, it is still an important artifact which belongs in a museum. Clearly an expert is needed to recover it.

Famous archaeologist and adventurer Indiana Jones has taken this dangerous task onto himself and traveled to Uruguay to find the cup. His search has led him to an ancient underground cave system where the cup is rumoured to have been hidden. Many traps lure within these caves, and only his instinct and his faithful whip have saved Indy from certain death. Now he has reached a mysterious enormous gate and he can only speculate that the cup must be hidden behind that gate. Unfortunately it is shut close.

The gate is riddled with switches and levers, and all of them are denoted with letters and numbers. As you may have guessed, the gate will only open if the switches and levers are pulled in the correct order, but beware! - For if anyone is unlucky enough to get the order wrong, doom awaits him.

Luckily, during his exploration of the caves Indy has found several encrypted hints which provide clues about the correct sequence. Here's one: "The faithful knows that X comes before O" And another: "Under no circumstances should you touch Δ unless the Θ has been moved!" Clearly these clues give hints about the correct order, but there are a lot of switches and levers, and there are lots of clues. Indy needs help!

Given all of the hints Indy has collected, can you help him determine the correct order of the levers and switches so that he can successfully complete his adventure? But beware – Indy could have missed some hints; or perhaps he misinterpreted some of them. The former case will likely leave more than one possible sequence while the latter will lead to no possible sequence at all. You must detect these cases and warn Indy.

Input

The first line of input contains the number of test cases C ($C \leq 30$). For each test case there is one line with the number n ($1 \leq n \leq 10\,000$) of switches/levers on the gate and the number h ($0 \leq h \leq 100\,000$) of hints that Indy has discovered. Then follow h lines, one for each clue, with the numbers a and b ($1 \leq a, b \leq n, a \neq b$), meaning that lever a must be pulled before lever b.

Output

For each test case output one line with the correct sequence of the numbers 1 to n. Separate the numbers with a whitespace. If there is no possible sequence, print "`recheck hints`" instead. If there are multiple possible sequences, print "`missing hints`" instead.

Sample Input

```
3
3 2
1 2
3 1
3 1
1 2
3 2
1 2
2 1
```

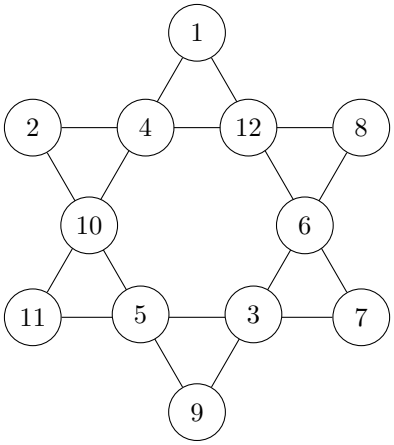
Sample Output

```
3 1 2
missing hints
recheck hints
```

This page is intentionally left (almost) blank.

Problem D
Magic Star

A magic star consists of all the numbers from 1 to 12 arranged in the shape of a hexagram:



The magic comes from the fact that in each line of 4 numbers, the sum of the numbers is 26. In the example given above, the six lines consist of the following numbers:

- $1 + 4 + 10 + 11$
- $11 + 5 + 3 + 7$
- $7 + 6 + 12 + 1$
- $2 + 10 + 5 + 9$
- $9 + 3 + 6 + 8$
- $8 + 12 + 4 + 2$

There are several possible ways to arrange the numbers to get a magic star. Given a partially labelled star, your task is to extend the solution such that a magic star is formed.

Input

The input consists of a visualization of the star; the unlabelled fields of the star will be represented by an 'x' character, and labelled fields will contain a letter between 'A' and 'L', where the i-th letter in the alphabet represents number i. The character '.' is used to align the fields of the star in the shape of a hexagram. You may assume that each input will use the same alignment of the fields as the one in the sample input.

Output

Print the lexicographically smallest extension of the given partial solution which is a magic star (lexicographically smallest means that the concatenation of the rows should result in a string which is lexicographically smaller than other potential solutions). You may assume that there is always a solution for the given input.

Sample Input

```
....x....
.A.I.D.x.
..x...x..
.x.x.x.x.
....x....
```

Sample Output

```
....F....
.A.I.D.L.
..H...E..
.C.J.B.K.
....G....
```

This page is intentionally left (almost) blank.

Problem E
Magical Crafting

One of the recent hypes in terms of independent games has been Minecraft, a cute little game in which you mine some materials and are able to craft a lot of stuff using these materials. Since you already mastered the Minecraft universe, you decide to write your own little mod for the game to add magical crafting.

For a magic craft, you start with a single block of magic stone $m_1 =$ 'A'. By adding a specific amount of diamonds you can transform the magic stone into two magic stones $m_2 \in$ 'A' ... 'Z' and $m_3 \in$ 'A' ... 'Z', or even into a non-transformable final glow stone g. This is called a crafting recipe. The glow stone transformation can only transform a magic stone m into its matching lighting effect g, i.e. the magic stone 'A' can be transformed in the lighting effect 'a'. This transformation always costs exactly 1 diamond. This way you want to be able to craft wonderful magic lights for your own world.

But as you are aware of the rareness of the precious diamonds, and also very picky about the lighting effects you want to achieve, you have to test your crafting recipes. The question arises if your set of crafting recipes allows you to create all your desired lighting effects and how many diamonds you will need.

Input

The first line of the input gives the number of test cases C ($0 < C \leq 100$). Each test case starts with two integers $0 \leq R \leq 30$, $0 \leq L \leq 10$ on a single line, denoting the set of recipes and the number of lighting effects. The next R lines contain the set of recipes. Each crafting recipe is given by the magic stones m_1, m_2, and m_3 as well es the number of diamonds necessary $0 \leq d \leq 1\,000$ on a single line. The next L lines contain the lighting effects you want to achieve. Each of theses lines contains an integer $1 \leq l \leq 100$, which represents the length of the lighting effect. The line completes with a single string of length l, consisting only of lower case characters $a \ldots z$, the glowstones within the effect.

Output

For every test case print "`CASE #`" followed by the number of the test case, starting with 1, on a single line. For each lighting effect print a single line. If the effect is possible print "`POSSIBLE WITH X DIAMONDS`", with X representing the number of diamonds you will need to achieve the effect. In the case the lighting effect cannot be crafted, print "`IMPOSSIBLE`".

Sample Input

```
2
1 4
A A A 2
3 aaa
8 aaaaaaaa
5 ababa
1 a
3 1
A B C 1
C P C 7
B I C 11
4 icpc
```

Sample Output

```
CASE #1
POSSIBLE WITH 7 DIAMONDS
POSSIBLE WITH 22 DIAMONDS
IMPOSSIBLE
POSSIBLE WITH 1 DIAMONDS
CASE #2
POSSIBLE WITH 23 DIAMONDS
```

This page is intentionally left (almost) blank.

Problem F

My brother's diary

Nowadays, people who want to communicate in a secure way use asymmetric encryption algorithms such as RSA. However, my older brother uses another, simpler encryption method for his diary entries. He uses a substitution cipher where each letter in the plaintext is substituted by another letter from the alphabet. The distance between the plaintext letter and the encrypted letter is fixed. If we would define this fixed distance d to 5, A would be replaced by F, B by G, C by H, ..., Y by D, Z by E.

With a fixed and known distance d the decryption would be somewhat simple. But my brother uses random distances for each of his diary entries. To decrypt his diary I have to guess the distance d for each entry. Thus, I use the well known phenomenon that the letter E is used more often in English words than other letters.

Can you write a program for me that calculates the distance d based on the fact that the most used letter in the encrypted text corresponds to the letter E in plaintext? Of course, I am interested in the decrypted text, too.

Input

The input consists of several test cases c that follow ($1 \leq c \leq 100$). Each test case is given in exactly one line containing one diary entry. Diary entries only use upper case letters (A-Z) and spaces. Each diary entry consists of at most 1 000 encrypted letters (including spaces).

Output

For each test case, print one line containing the smallest possible distance d ($0 \leq d \leq 25$) and the decrypted text. If the decryption is not possible because there are multiple distances conforming to the rules above, print "`NOT POSSIBLE`" instead. Spaces are not encrypted.

Sample Input

```
4
RD TQIJW GWTYMJWX INFWD JSYWNJX ZXJ F XNRUQJ JSHWDUYNTS YJHMSNVZJ
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
XVIDRE TFCCVXZRKV GIFXIRDDZEX TFEKVJK UVTIPGKZFE
XVIDRE TFCCVXZRKV GIFXIRDDZEX TFEKVJK
```

Sample Output

```
5 MY OLDER BROTHERS DIARY ENTRIES USE A SIMPLE ENCRYPTION TECHNIQUE
10 JXU GKYSA RHEMD VEN ZKCFI ELUH JXU BQPO TEW
17 GERMAN COLLEGIATE PROGRAMMING CONTEST DECRYPTION
NOT POSSIBLE
```

This page is intentionally left (almost) blank.

Problem G

Security Zone

The manager of a large security company has to build a new surveillance system for different sites. A site consists of N objects where each object has its own security circle. At the border of the surveillance system a high voltage fence has to be installed. The security zone inside of the fence has to be connected. Furthermore, all objects and their security circles should be inside the security zone. The security circles of different objects will never overlap or touch. Now the manager needs your help. He asks you for the minimal needed fence length.

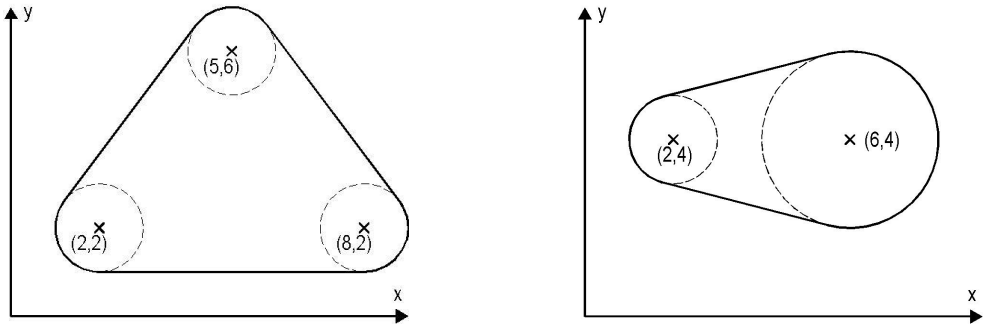


Figure 1 – Illustration of first and second sample input.

Input

The first line of the input gives the number of test cases C ($0 \leq C \leq 100$). The first line of each such test case holds the integer N: the number of objects in the current site ($0 < N \leq 25$). Each of the following N lines holds three integers x_i, y_i, and r_i that describe an object. The coordinates of the i-th object and the radius for the needed security circle for this object. ($|x_i|, |y_i| \leq 100$, $0 < r_i \leq 100$) (The center of the security circle around an object is the position of the object itself.)

Output

For each test case print one line containing the minimal fence length for this case. Your output should have an absolute or relative error of at most 10^{-7}.

Sample Input	Sample Output
3 3 2 2 1 8 2 1 5 6 1 2 6 4 2 2 4 1 4 2 2 2 6 1 1 5 5 2 1 6 1	22.2831853072 17.6761051635 25.4247779608

This page is intentionally left (almost) blank.

Problem H
Sightseeing

As a computer science student you are of course very outdoorsie, so you decided to go hiking. For your vacation this year, you located an island full of nice places to visit. You already identified a number of very promising tracks, but are still left with some problems. The number of choices is so overwhelming, that you had to select only a "small" subset of at most 100 000 sights. And if that is not enough, you are very picky about the order in which you want to visit the sights. So you have already decided on an order in which you want to visit the preselected tracks. The problem you are left with is to decide in which direction to travel along each single track, and whether you may have to reduce your choice of tracks even further. After identifying the travel time between the endpoints of different tracks, you decide to write a program to figure out if you can make all your trips within the time you have planned for your vacation. Since you also do not want to waste any precious time, you only care about an optimal solution to your problem. Furthermore, the tracks can get pretty challenging. Thats why you do not want to hike along a track more than once.

Input

The first line of the input gives the number of test cases C ($0 < C \leq 100$). The first line of each such test case holds two integers N, T the number of tracks of the current hiker ($1 \leq N \leq 100\,000$) and the maximal time spent hiking throughout the vacation ($0 \leq T \leq 1\,000\,000$) Each of the following N lines holds five integers c_p, c_{bb}, c_{be}, c_{eb} and c_{ee} that describe a track (in order of importance). c_p gives the length of the track in minutes. c_{xy} gives the travel time of the official **b**egin or **e**nd of a track to the **b**eginning or **e**nd of the next most important track, where x and y are either b or e. All values given are non-negative integers not greater than 1 000 000. Since you have to get back to your car, the list is circular. Furthermore, we will ignore the time it takes you to get to the start of your trip with your car.

Output

For each test case print one line. The output should contain a list of "`F`" and "`B`" for every track (in order) indicating whether you have to hike the track in forward direction or backward direction. If you cannot make the full trip within the planned time T, you should print "`IMPOSSIBLE`" to indicate that these trips are just too much hiking.

Sample Input

```
3
2 100
4 7 8 2 3
1 4 6 1 2
2 20
4 2 3 7 8
1 1 2 4 6
3 5
1 2 2 2 1
1 1 2 2 2
1 2 2 1 2
```

Sample Output

```
FF
BB
IMPOSSIBLE
```

This page is intentionally left (almost) blank.

Problem I
Suiting Weavers

Willy the Weaver is in desperate hope to get married to the most beautiful and delightful female weaver Wilmar. Of course, Willy is not the only weaver interested in Wilmar.
In order to impress the females, weavers build elaborately woven nests using leaf fibers. Tomorrow is the big day on which Wilmar will inspect all nests. There is a heavy storm at the moment, and no weaver can leave his nest within the hours before sunrise. However, the storm will produce many piles of leaf fibers, so that all weavers will have a chance to improve their nests. Therefore, Willy is curious if he can succeed weaving the most impressive nest, so that Wilmar will finally decide on getting married to him. Since size matters, Willy tries to figure out how large his nest and the ones of his rivals may become.
For this purpose, Willy takes into consideration all known places offering leaf fibers suitable for nest construction. Since weavers do not like to leave their known territory, many of these places can be accessed by a subset of all weavers only, and some might even not be reachable by any weaver.
To reduce complexity, Willy does not want to set up a flight plan. This implies that he does not consider any particular strategy of his rivals nor does he make any assumptions on how many fibers they can carry at a time or how quick and when they fly. It is therefore possible that a weaver succeeds in picking up all fibers in his territory. Finally, Willy assumes that all weavers are as integer as he is: they do not steal fibers from nests of their rivals.
Is there any chance that no weaver will have a larger nest (number of fibers) than Willy after all leaf fibers have been picked up?

Input

The first line contains the number of testcases T ($1 \leq T \leq 100$).
Each test case starts with a line containing two integers. The first integer W ($1 \leq W \leq 100$) is the number of weavers (including Willy); the second one P ($1 \leq P \leq 400$) is the number of places with leaf fibers.
Next come W lines describing the nest of each weaver with four integers x, y, f, and r ($0 \leq x, y, r \leq 10\,000$, $1 \leq f \leq 10\,000$): x and y define the position of the nest, f is the size of the nest in number of fibers, and r is the radius of the territory in which the owner of the nest will search for additional fibers. The first of these W lines describes Willy's nest.
Thereafter follow P lines defining the places with available leaf fibers with three integers x, y, and f ($0 \leq x, y \leq 10\,000$, $1 \leq f \leq 10\,000$): x and y define the position of the place, and f is the number of available leaf fibers.

Output

For each test case print one single line containing the string `Suiting Success`, if Willy has a chance to marry Wilmar after all fibers have been picked up (a tie in nest size is sufficient); else print `Lonesome Willy`.

Sample Input

```
2
3 2
0 0 1 10
10 0 1 10
20 0 1 10
5 0 2
15 0 4
3 2
0 0 1 10
10 0 1 10
20 0 1 10
5 0 2
15 0 5
```

Sample Output

```
Suiting Success
Lonesome Willy
```

This page is intentionally left (almost) blank.

Problem J

Time to live

As you might know, most computer networks are organized in a tree-like fashion, i.e. each computer is reachable by each other computer but only over one, unique path.
The so-called *Time to live* (TTL) specifies after how many hops a network packet is dropped if it has not reached its destination yet. The purpose of the TTL is to avoid situations in which a packet circulates through the network caused by errors in the routing tables.
The placement of a router that connects the network to another network is optimal when the maximal needed TTL for packets that are sent from this router to any other computer within the same network is minimal. Given a network as specified above, you should calculate the maximal needed TTL in this network if you can select the computer that should be used as router.

Input

The first line of the input consists of the number of test cases c that follow ($1 \leq c \leq 100$). Each test case starts with a line specifying N, the number of computers in this network ($1 < N \leq 100\,000$). Computers are numbered from 0 to $N - 1$. Then follow $N - 1$ lines, each specifying a network connection by two numbers a and b which means that computer a is connected to computer b and vice versa, of course ($0 \leq a, b < N$).

Output

For each test case in the input, print one line containing the optimal TTL as specified above.

Sample Input	Sample Output
3 2 1 0 5 3 2 2 1 0 2 2 4 9 3 1 6 5 3 4 0 3 8 1 1 7 1 6 2 3	1 1 2

This page is intentionally left (almost) blank.