I
Regular Convex Polygon

A regular convex polygon is a polygon where each side has the same length, and all interior angles are equal and less than 180 degrees. A square, for example, is a regular convex polygon. You are given three points which are vertices of a regular convex polygon R; can you determine the minimum number of vertices that R must have?

Input

Each test case consists of three lines. Line i consists of two floating point values x_i and y_i ($-10^4 \leq x_1, y_1 \leq 10^4$) where (x_i, y_i) are the coordinates of a vertex of R. The coordinates are given with a precision of 10^{-6}, i.e., they differ from the exact coordinates by at most 10^{-6}. You may assume that for each test case the Euclidean distance between any two given points is at least 1, and R has at most 1000 vertices. The input will finish with a line containing the word `END`.

Output

For each test case, print one line with the minimum number of vertices that R must have.

Sample Input

```
-1385.736326 -146.954822
430.000292 -2041.361203
1162.736034 478.316025
0.000000 4147.000000
-4147.000000 0.000000
0.000000 -4147.000000
END
```

Sample Output

```
3
4
```

A

G Smoking gun

Andy: "Billy the Kid fired first!"

Larry: "No, I'm sure I heard the first shot coming from John!"

The arguments went back and forth during the trial after the big shoot-down, somewhere in the old wild west. Miraculously, everybody had survived (although there were serious injuries), but nobody could agree about the exact sequence of shots that had been fired. It was known that everybody had fired at most one shot, but everything had happened very fast. Determining the precise order of the shots was important for assigning guilt and penalties.

But then the sheriff, Willy the Wise, interrupted: "Look, I've got a satellite image from the time of the shooting, showing exactly where everybody was located. As it turns out, Larry was located much closer to John than to Billy the Kid, while Andy was located just slightly closer to John than to Billy the Kid. Thus, because sound travels with a finite speed of 340 meters per second, Larry may have heard John's shot first, even if Billy the Kid fired first. But, although Andy was closer to John than to Billy the Kid, he heard Billy the Kid's shot first – so we know for a fact that Billy the Kid was the one who fired first!

Your task is to write a program to deduce the exact sequence of shots fired in situations like the above.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers n ($2 \leq n \leq 100$) and m ($1 \leq m \leq 1\,000$): the number of people involved and the number of observations.
- n lines with a string S, consisting of up to 20 lower and upper case letters, and two integers x and y ($0 \leq x, y \leq 1\,000\,000$): the unique identifier for a person and his/her position in Cartesian coordinates, in metres from the origin.
- m lines of the form "`S1 heard S2 firing before S3`", where $S1$, $S2$ and $S3$ are identifiers among the people involved, and $S2 \neq S3$.

If a person was never mentioned as $S2$ or $S3$, then it can be assumed that this person never fired, and only acted as a witness. No two persons are located in the same position.

The test cases are constructed so that an error of less than 10^{-7} in one distance calculation will not affect the output.

Output

Per test case:

- one line with the ordering of the shooters that is compatible with all of the observations, formatted as the identifiers separated by single spaces.

If multiple distinct orderings are possible, output "`UNKNOWN`" instead. If no ordering is compatible with the observations, output "`IMPOSSIBLE`" instead.

Sample in- and output

Input	Output
3 4 2 BillyTheKid 0 0 Andy 10 0 John 19 0 Larry 20 0 Andy heard BillyTheKid firing before John Larry heard John firing before BillyTheKid 2 2 Andy 0 0 Beate 0 1 Andy heard Beate firing before Andy Beate heard Andy firing before Beate 3 1 Andy 0 0 Beate 0 1 Charles 1 3 Beate heard Andy firing before Charles	BillyTheKid John IMPOSSIBLE UNKNOWN

C

Problem A — limit 5 seconds

Good or Bad?

Description

Bikini Bottom has become inundated with tourists with super powers. Sponge Bob and Patrick are trying to figure out if a given character is good or bad, so they'll know whether to ask them to go jelly-fishing, or whether they should send Sandy, Mermaid Man, and Barnacle Boy after them.

SPONGE BOB: Wow, all these characters with super powers and we don't know whether they are good guys or bad guys.
PATRICK: Well, it's easy to tell. You just have to count up the number of g's and b's in their name. If they have more g's, they are good, if they have more b's, they are bad. Think about it, the greatest hero of them all, Algorithm Crunching Man is good since he has two g's and no b's.
SPONGE BOB: Oh, I get it. So Green Lantern is good and Boba Fett is bad!
PATRICK: Exactly! Uh, who's Boba Fett?
SPONGE BOB: Never mind. What about Superman?
PATRICK: Well he has the same number of g's as b's so he must be neutral.
SPONGE BOB: I see, no b's and no g's is the same number. Very clever Patrick! Well what about Batman? I thought he was good.
PATRICK: You clearly never saw *The Dark Knight*...
SPONGE BOB: Well what about Green Goblin? He's a baddy for sure and scary!
PATRICK: The Green Goblin is completely misunderstood. He's tormented by his past. Inside he's good and that's what counts. So the method works!
SPONGE BOB: Patrick, you are clearly on to something. But wait, are you saying that Plankton is neutral after all the terrible things he's tried to do to get the secret Crabby Patty formula?
PATRICK: Have any of his schemes ever worked?
SPONGE BOB: Hmmm, I guess not. Ultimately he's harmless and probably just needs a friend. So sure, neutral works for him.
PATRICK: Alright then, let's start taking names and figure this out.
SPONGE BOB: But Patrick, if we start counting all day, Squidward will probably get annoyed and play his clarinet and make us lose count.
PATRICK: Well, let's hire a human to do it for us on the computer. We'll pay them with Crabby Patties!
SPONGE BOB: Great idea Patrick. We're best friends forever!

Help Sponge Bob and Patrick figure out who is good and who is bad.

Input

The first line will contain an integer n ($n > 0$), specifying the number of names to process. Following this will be n names, one per line. Each name will have at least 1 character and no more than 25. Names will be composed of letters (upper or lower case) and spaces only. Spaces will only be used to separate multiple word names (e.g., there is a space between Green and Goblin).

Output

For each name read, display the name followed by a single space, followed by " is ", and then followed by either "GOOD", "A BADDY", or "NEUTRAL" based on the relation of b's to g's. Each result should be ended with a newline.

Sample Input	Sample Output
8 Algorithm Crunching Man Green Lantern Boba Fett Superman Batman Green Goblin Barney Spider Pig	Algorithm Crunching Man is GOOD Green Lantern is GOOD Boba Fett is A BADDY Superman is NEUTRAL Batman is A BADDY Green Goblin is GOOD Barney is A BADDY Spider Pig is GOOD

D: Vive la Difference!

Take any four positive integers: **a**, **b**, **c**, **d**. Form four more, like this:

|a-b| |b-c| |c-d| |d-a|

That is, take the absolute value of the differences of **a** with **b**, **b** with **c**, **c** with **d**, and **d** with **a**. (Note that a zero could crop up, but they'll all still be non-negative.) Then, do it again with these four new numbers. And then again. And again. Eventually, all four integers will be the same. For example, start with `1,3,5,9`:

```
1 3 5 9
2 2 4 8 (1)
0 2 4 6 (2)
2 2 2 6 (3)
0 0 4 4 (4)
0 4 0 4 (5)
4 4 4 4 (6)
```

In this case, the sequence converged in 6 steps. It turns out that in all cases, the sequence converges very quickly. In fact, it can be shown that if all four integers are less than 2^n, then it will take no more than 3*n steps to converge!

Given **a**, **b**, **c** and **d**, figure out just how quickly the sequence converges.

Input

There will be several test cases in the input. Each test case consists of four positive integers on a single line (1 ≤ **a**,**b**,**c**,**d** ≤ 2,000,000,000), with single spaces for separation. The input will end with a line with four 0s.

Output

For each test case, output a single integer on its own line, indicating the number of steps until convergence. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
1 3 5 9
4 3 2 1
1 1 1 1
0 0 0 0
```

Sample Output

```
6
4
0
```

D

H
Peer Review

For scientific conferences, scientists submit papers presenting their ideas, and then review each other's papers to make sure only good papers are presented at the conference. Each paper must be reviewed by several scientists, and scientists must not review papers written by people they collaborate with (including themselves), or review the same paper more than once.

You have been asked to write a program to check if your favorite conference is doing things right. Whether a paper is being reviewed too much, too little, or by the wrong people - the organizers must know before it is too late!

Input

The first line in each test case has two integers, K ($1 \leq K \leq 5$) and N ($1 \leq N \leq 1000$). K is the number of reviews that each paper will receive, while N is the number of papers to be reviewed. The conference only accepts papers with a single author, and authors can only present a single paper at the conference.

Each of the next N lines describes an author and includes the name of the institution to which the author belongs, followed by the list of the K papers he or she has been requested to review. It is assumed that researchers from the same institution collaborate with each other, whereas researchers from different institutions don't. All institution names are shorter than 10 characters, and contain only upper or lowercase letters and no whitespace. Since we have as many papers as authors, papers are identified by their author's index; paper 1 was written by the first author in the list, and paper N was written by the last author.

The end of the test cases is marked with a line containing $K = 0$ and $N = 0$. You should generate no output for this line.

Output

For each test case, your program should output `NO PROBLEMS FOUND` (if all rules are being followed) or P `PROBLEMS FOUND`, where P is the number of rule violations found (counting at most 1 violation per paper). If there is exactly one rule violation overall, your program should output `1 PROBLEM FOUND`.

Sample Input

```
2 3
UCM 2 3
UAM 1 3
UPM 1 2
2 3
UCM 2 3
UAM 1 2
UPM 2 2
0 0
```

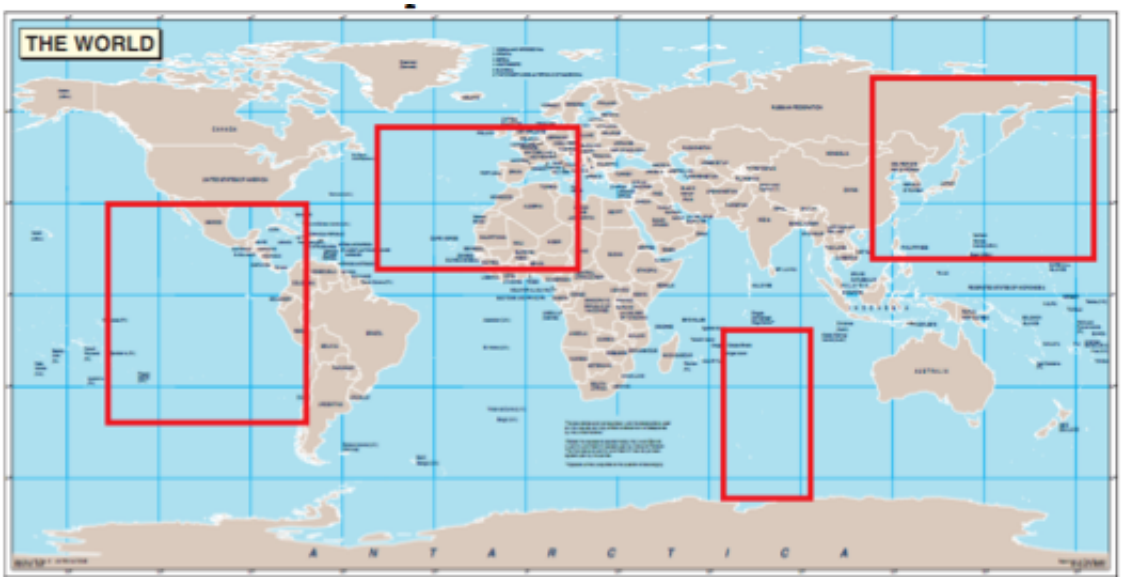
Sample Output

```
NO PROBLEMS FOUND
3 PROBLEMS FOUND
```

E

6 - Thinking Inside the Box

Data Co is having trouble organizing its vast collection of data. It has dozens of sets of survey data that span large portions of the globe. The bounding boxes of these surveys are stored so that anyone curious about a specific region's geological information merely needs to enter in some coordinates to retrieve the data; and that's where you come in...



The input file contains two lists, with each entry separated by a line break. The first is a list of bounding boxes stored in the system (These will be referred to as “Data Boxes”). The second is a list of bounding boxes that represent user queries (“Query Boxes”). Each list is preceded by the number of items in that list.

Bounding boxes are represented by two Latitude/Longitude coordinates. The first represents a top corner of the box, the second represents the opposite corner of the box. The two points are separated by a space.

The program must determine which Data Boxes are touched in any way (sharing borders counts) by Query Boxes. The program should print out all Data Boxes touched by Query Boxes. If multiple Query Boxes touch the same Data Box, the coordinates of the Data Box must only be printed out once.

If no "touching" data and query boxes are found print "`No data found.`" without the quotes.

Tips:

Latitude and Longitude are angular measurements. Lines of Latitude are the horizontal lines on the globe. The Equator has a Latitude of `0°`, the North Pole has a Latitude of `90°`, and the South Pole has a Latitude of `-90°`. Lines of Longitude, called ‘Meridians’ wrap vertically around the globe, always starting and ending at the poles. The Prime Meridian, which passes through Greenwich England, has a Longitude of `0°`. Longitude values ‘wrap around’ at the opposite Meridian at `±180`. Longitude decreases as you go west of the Prime Meridian until it ‘wraps around’ at `-180` (for example, New Orleans is at Longitude `-90`). Longitude increases as you go east of the Prime Meridian, again until it wraps around at `180` (Beijing is at Longitude `116`).

Both Query and Data Boxes can wrap ‘around the middle’ of the globe, but may not wrap ‘over the top’ o
the globe.

F

Sample Input	Sample Output
2 3 61,34 47,97 35,-65 24,-27 -5,-124 -46,-88 2 45,-128 -45,-5 -8,67 -42,82 4 1,1 -90,160 45,-160 -5,-155 82,-42 67,-8 60,0 10,90 5 -5,-155 -50,-130 90,-90 25,-30 80,-50 75,-20 50,180 40,-110 -65,55 -80,80	35,-65 24,-27 -5,-124 -46,-88 1,1 -90,160 45,-160 -5,-155 82,-42 67,-8

Greater New York Programming Contest
Adelphi University
Garden City, NY

event sponsors

A • Repeating Characters

For this problem, you will write a program that takes a string of characters, ***S***, and creates a new string of characters, ***T***, with each character repeated ***R*** times. That is, ***R*** copies of the first character of ***S***, followed by ***R*** copies of the second character of ***S***, and so on. Valid characters for ***S*** are the *QR Code* "*alphanumeric*" characters:

```
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ$%*+-./:
```

Input

The first line of input contains a single integer ***P***, ($1 \le P \le 1000$), which is the number of data sets that follow. Each data set is a single line of input consisting of the data set number ***N***, followed by a space, followed by the repeat count ***R***, ($1 <= R <= 8$), followed by a space, followed by the string ***S***. The length of string ***S*** will always be at least one and no more than 20 characters. All the characters will be from the set of characters shown above.

Output

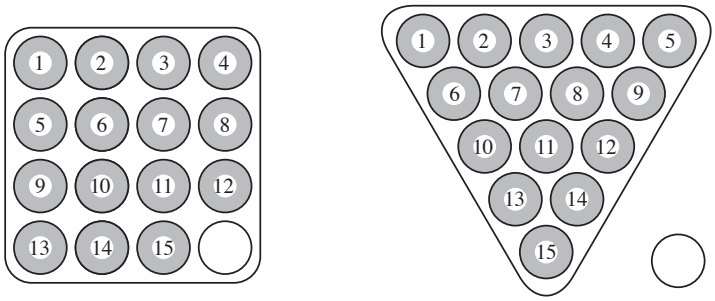
For each data set there is one line of output. It contains the data set number, ***N***, followed by a single space which is then followed by the new string ***T***, which is made of each character in ***S*** repeated ***R*** times.

Sample Input	Sample Output
2 1 3 ABC 2 5 /HTP	1 AAABBBCCC 2 /////HHHHHTTTTTPPPPP

G

Problem D: Everyone out of the Pool

When you rent a table at a pool hall, the proprietor gives you a 4-by-4 tray of 16 balls, as shown in Figure (a) below. One of these balls, called the "cue ball", is white, and the remaining 15 are numbered 1 through 15. At the beginning of a game, the numbered balls are racked up in a triangle (without the cue ball), as shown in Figure (b).



(a) (b)

Now imagine other pool-like games where you have a cue ball and x numbered balls. You'd like to be able to rack up the x numbered balls in a triangle, and have all $x+1$ balls perfectly fill a square m-by-m tray. For what values of x is this possible? In this problem you'll be given an lower bound a and upper bound b, and asked how many numbers within this range have the above property.

Input

Input for each test case will one line containin two integers a b, where $0 < a < b \leq 10^9$. The line 0 0 will follow the last test case.

Output

For each test case one line of output as follows:

`Case` n: k

if there are k integers x such that $a < x + 1 < b$, x balls can be racked up in a triangle, and $x + 1$ balls fill a square tray.

Sample Input

```
15 17
14 16
1 20
0 0
```

Sample Output

```
Case 1: 1
Case 2: 0
Case 3: 2
```

H

I

E Please, go first

You are currently on a skiing trip with a group of friends. In general, it is going well: you enjoy the skiing during the day and, of course, the après-skiing during the night. However, there is one nuisance: the skiing lift. As always, it is too small, and can only serve one person every 5 seconds. To make matters worse, you and your friends generally don't arrive simultaneously at the lift, which means that you spend time waiting at the bottom of the mountain for the lift and at the top again for your friends.

The waiting at the top is especially inefficient. In fact, you realize that if your friends haven't arrived yet, you might as well let other people pass you in the queue. For you, it makes no difference, since otherwise you'd be waiting at the top. On the other hand, your actions might save them time if their friends have already arrived and are currently waiting for them at the top.

You are wondering how much time would be saved if everybody adopts this nice attitude. You have carefully observed the queue for a while and noticed which persons form groups of friends. Suppose someone lets another pass if doing this doesn't change his own total waiting time, but saves time for the other person. Do this over and over again until it can't be done anymore. How much time will this save, in total?

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($1 \leq n \leq 25\,000$): the number of people in the line for the lift.
- one line with n alphanumeric characters (uppercase and lowercase letters and numbers): the queue. The first person in this line corresponds to the person at the head of the queue. Equal characters correspond to persons from the same group of friends.

Output

Per test case:

- one line with an integer: the time saved, in seconds.

Sample in- and output

Input	Output
2 6 AABABB 10 Ab9AAb2bC2	15 45

I