Problem A
Count
Problem ID: count

This problem is strictly to acclimate teams to the contest environment. We strongly suggest you first finish this problem, and then attempt the more complex practice problem.

Input

The input file will contain an unknown number of lines with at most 100 characters on each line. All the characters will be printable ASCII characters. Note that the input file will always exist, but may be empty.

Output

The number of lines in the input file.

Sample Input	Sample Output
one two	2

Problem B: Fuel Stops

Source: `fuel.{c,cpp,java}`

You are required to take a circular tour of a given set of cities: start at a certain city, visit each city once, and return to the city at which you started.

Each city is identified by a number: 1, 2, 3, etc. The numbering of the cities specifies the path you must take, but the starting point is not specified. From the highest numbered city, you proceed to City 1. For example, if there are three cities (numbered 1, 2, 3) you have three choices for completing the tour:

Start at 1, proceed to 2, then to 3, then return to 1.
Start at 2, proceed to 3, then to 1, then return to 2.
Start at 3, proceed to 1, then to 2, then return to 3.

There is a refueling station in each city, with a given quantity of available fuel. The sum of all the fuel supplies at the refuelling stations is equal to the fuel required to make the entire tour. You start with an empty tank at one of the refuelling stations. You will be running out of fuel just as you pull into the starting station upon successful completion of the tour.

You must determine which city (or cities) will qualify as a starting point for the tour without running out of fuel before returning to the starting station. Assume the fuel tank is sufficiently large to handle all of the refuelling operations.

Input

The input will contain data for several test cases. For each test case, there will be three lines of data. The first line will specify the number of cities as a single integer. The second line will specify the quantity of fuel available at each of the refuelling stations, in the order of city numbers: 1, 2, 3, etc. The third line will specify the quantity of fuel needed to get from each station to the next one, in the order of city numbers: from the station at city 1 to the station at city 2, from the station at city 2 to the station at city 3, etc; the last number specifies the quantity of fuel required to get from the highest numbered city's station back to the station at city 1. All fuel quantities are positive integers given in imperial gallons. The sum of the fuel supplies will not exceed the range of signed 32-bit integers. There will be at least two cities and up to 100000 cities. End of input will be indicated by a line containing zero for the number of cities; this line will not be processed.

Output

For each test case, there will be one line of output. After the case number, the output will list the city numbers that work as starting cities for a successful tour, as described above. In case of several possible starting cities, they must be listed in increasing order separated by a single space. Follow the format of the sample output. The Hungarian mathematician L. Lovász proved that there is always at least one possible starting city.

Sample Input

```
3
3 2 2
4 2 1
3
3 2 1
1 3 2
4
3 4 5 2
2 3 8 1
0
```

Sample Output

```
Case 1: 2 3
Case 2: 1
Case 3: 4
```

Problem C: Magical GCD

The *Magical GCD* of a nonempty sequence of positive integers is defined as the product of its length and the greatest common divisor of all its elements.

Given a sequence $(a_1, ..., a_n)$, find the largest possible Magical GCD of its connected subsequence.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with a line containing a single integer n, $1 \leqslant n \leqslant 100\,000$. The next line contains the sequence $a_1, a_2, ..., a_n$, $1 \leqslant a_i \leqslant 10^{12}$.

Output

For each test case output one line containing a single integer: the largest Magical GCD of a connected subsequence of the input sequence.

Example

For an example input	the correct answer is:
1 5 30 60 20 20 20	80

Problem D: Subway

Johny is going to visit his friend Michelle. His dad allowed him to go there on his own by subway. Johny loves traveling by subway and would gladly use this opportunity to spend half a day underground, but his dad obliged him to make as few line changes as possible. There are a lot of stations in the city, and several subway lines connecting them. All trains are perfectly synchronized – the travel between two consecutive stations on every line takes exactly one minute, and changing lines at any station takes no time at all.

Given the subway map, help Johny to plan his trip so that he can travel for as long as possible, while still following his dad's order.

Input

First line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with an empty line. The next two lines begin with the strings **Stops:** and **Lines:**, and contain the names (separated by a comma and a space) of all subway stops and lines, respectively. A single line for each subway line follows (in no particular order), beginning with <**line-name**> **route:** and listing the names of the stops along this particular line. The final two lines specify the names of the (different) stations nearby Johny's and Michelle's homes.

In each test case, there are at most 300 000 stations and 100 000 lines, whose total length does not exceed 1 000 000. The names of lines and stations are between 1 and 50 characters long and can contain letters, digits, hyphens (**-**), apostrophes (**'**) and "and" signs (**&**). All lines are bidirectional (although changing the direction of travel counts as a line change) and there are no self-crossings.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case, print a single line summarizing the optimal route Johny can take (see example output for exact format). You may assume that such a route always exists.

Example

Some lines in the example test data below were too long and had to be wrapped. You can access full sample tests at your workstation.

For an example input

```
3

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at King'sCross
Michelle lives at GreenPark

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at PiccadillyCircus
Michelle lives at LeicesterSquare

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at Victoria
Michelle lives at HydeParkCorner
```

the correct answer is:

```
optimal travel from King'sCross to GreenPark: 1 line, 3 minutes
optimal travel from PiccadillyCircus to LeicesterSquare: 1 line, 1 minute
optimal travel from Victoria to HydeParkCorner: 2 lines, 7 minutes
```

Problem E: Chain & Co.

Chain & Co. specializes in producing infinitely strong chains. Because of their high quality products, they are quickly gaining market share. This leads to new challenges, some of which they could have never imagined before. Like, for example, automatic verification of link endurance with a computer program, which you are supposed to write.

The company produces *links* of equal size. Each link is an infinitely thin square frame in three dimensions (made of four infinitely thin segments).

During tests all links are axis-aligned [1] and placed so that no two frames touch. To make a proper strength test, two sets of links A and B are forged so that every link of A is inseparable from every link of B (being inseparable means that they cannot be moved apart without breaking one of them).

You stumble upon some links (axis-aligned, pairwise disjoint). Are they in proper testing position? In other words, can they be divided into two non-empty sets A and B with the desired property?

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with an empty line. The next line contains an integer n, $1 \leqslant n \leqslant 10^6$ – the number of links in the chain. Each of the next n lines contains 6 space-separated integers $x_i, y_i, z_i, x'_i, y'_i, z'_i$, all between -10^9 and 10^9 – the coordinates of two opposite corners of the i-th link.

Output

For each test case, print a single line containing the word **YES** if the set is in proper testing position, or **NO** otherwise.

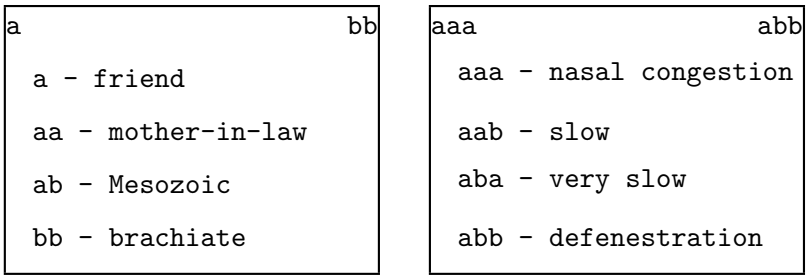
[1] Axis-aligned means that all segments are parallel to either X, Y, or Z axis.

Example

For an example input	the correct answer is:
3 2 0 0 0 0 10 10 -5 5 15 5 5 25 5 0 0 0 0 10 10 -5 5 6 5 5 16 -5 5 -6 5 5 4 -5 6 5 5 16 5 -5 -6 5 5 4 5 3 0 0 0 3 0 -3 1 -1 -1 1 2 -4 -1 -2 -2 2 1 -2	NO YES YES

Problem F: Xenospeak

It's 2014, and Bob Roberts is an expert linguist and world renowned expert on the language of the alien M'ca. Their language is rather unusual in that all the words are made up of the letter combinations "a", "ab" and "bb" (we're using 'a's and 'b's here, since the actual M'ca characters are unprintable). Thus, some words in their language are aaabbbbb and aababb, but babb is not (you'd be a laughingstock in any M'ca establishment if you tried to used babb in a sentence). Not surprisingly, with such a small alphabet, every possible combination of "a", "ab" and "bb" form a legal M'ca word (up to a certain length, which is of no importance to this problem). Bob is creating a set of M'ca-to-English dictionaries of all legal M'ca words and is following the traditional word ordering of the M'ca: all 1 letter words are listed first (in alphabetical order), then all 2-letter words (in alphabetical order), and so on. The first two pages of one possible dictionary are shown below:



Bob needs a little help. He intends for each page of any dictionary to contain the same number of M'ca words, but this number will vary in different editions depending on the page size, font size, etc. As with any dictionary, the first and last word of each page is printed at the top of the page to allow easier searching by users. Here's where you come in: given the number of words per page, and the page number, he would like a program to determine the two words printed at the top of that page.

Input

Input for each test case will consist of a single line containing two positive integers n m, where n is the number of words per page (≤ 30) and m is the page number ($m \leq 10^{18}$). A line containing 0 0 will terminate input.

Output

For each test case output the two words which would appear at the top of page m given that n words are printed on each page (including page m).

Sample Input

```
4 2
9 10
0 0
```

Sample Output

```
Case 1: aaa abb
Case 2: bbbbaa aaaabab
```