Problem A. Augmented Reality Game

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Augmented reality game players are acting on the map of the Earth capturing portals and linking them to each other using portal keys from their inventory. To link portals A and B, a player goes to portal B and uses a key to portal A to create a bidirectional link between them. This key to portal A is used up in the process and disappears from the player's inventory. A player may go to any portal at any time at his discretion. All keys to the same portal are indistinguishable.

When three portals are linked to each other, a triangular field is created. Fields can intersect each other arbitrarily. However, each field requires its own set of links, i. e. no fields can share the same link between portals. If two or more fields need to be created using portals A and B, a separate A–B link needs to be created for each field.

You are to help a player calculate the maximum number of fields he can create using a given set of keys he has.

Input

The first line of input contains an integer N: the number of distinct portals on the map ($1 \le N \le 10\,000$). Each of the next N lines contains one integer K_i: the number of keys a player has to portal i initially ($0 \le K_i \le 10\,000$).

Output

The first line of the output must contain a single integer F: the maximum number of fields that can be created using the given set of keys.

Examples

stdin	stdout
4 1 1 1 2	1
4 1 2 1 3	2

Problem B. Blacklist

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

In a certain online game, user IDs are represented as non-negative hexadecimal integers less than 10^{100}.

The administrator of the game received a message related to a hacker attack being planned. According to this message, hackers will use accounts with IDs which consist of pairwise distinct digits.

To prevent the attack, the administrator decided to put all such IDs into the blacklist. All possible IDs starting from 0 are processed sequentially. The IDs consisting of pairwise distinct digits are put into the blacklist. Processing all possible IDs is slow, and the administrator is in a hurry, so he asked you to solve the following problem: given the last processed ID (which can be put or not put into the blacklist), find the next ID that should be put into the blacklist.

Input

The input contains one integer P: the last processed ID written in hexadecimal form without leading zeroes ($0 \le P \le 2^{63} - 1$). Digits from A to F are represented by uppercase English letters from '`A`' to '`F`'.

Output

Print one integer written in hexadecimal form without leading zeroes: the next ID going to the blacklist. Use uppercase English letters from '`A`' to '`F`' to represent digits from A to F, respectively.

Examples

stdin	stdout
0	1
10	12
1FEE	2013

Problem C. Charisma and Skill

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Pavel Petrovich works as a coach in a ping-pong club in the "Rodina" palace of culture.

There are two brothers in the club: Artyom and Rodion Podavalov. Artyom trains a lot, and even though he could lose to anyone in the town in his first days, now he is the best and can beat everyone.

His brother Rodion had an advanced skill before, but his addiction to ACM contests prevailed over his aspiration for ping-pong and now he just spends all his free time at the university. So his sport abilities remain on the same level now.

But Artyom doesn't want to play in pair with anybody other than Rodion, in spite of talks that Rodion is spoiling his play. Artyom believes in numbers, so Pavel Petrovich has decided to use help of his friend Apollon Vladimirovich from the Polytechnic College to convince Artyom of bad influence of his partner. Apollon accepted the task with due responsibility and proposed a stochastic model. He proposed to measure the value of the ping-pong skill as a probability $0 \le p \le 1$ of victory over a random opponent.

The probability of victory for a pair also depends on the charisma relation of the players (second player over the first) $0 \le \alpha \le 1$ and equals $q = p_1 \cdot (1 - \alpha) + p_2 \cdot \alpha$, where p_1 and p_2 are the skill levels of the first and the second player — Artyom and Rodion respectively. Apollon suggested that the level of Artyom's skill in the first game was 0, in the last game it became 1, and it didn't decrease through the games. Rodion's skill as well as his relative charisma was constant.

Given the record of victories and defeats in a pair of these two players, you have to find these constant values using maximum likelihood estimation. More formally, you have to find values p_2 and α so that there exists a non-decreasing sequence of values of Artyom's skill starting from 0 and ending in 1, such that the product $P_1 \cdot P_2 \cdots P_N$ is maximized, where P_i is the probability of known outcome of i-th game. The product is maximized out of all possible values of $0 \le p_2, \alpha \le 1$ and all possible non-decreasing sequences of N values of Artyom's skill such that their first element is 0 and their last (N-th) element is 1.

Input

The first line of input contains one integer N: the number of games in the record ($2 \le N \le 10^6$). The second line contains N integers representing the outcomes of games: if i-th game has been lost, i-th number in the sequence is 0, otherwise it is 1.

Output

You have to output two numbers with absolute or relative precision 10^{-9}: the level of Rodion's skill and his relative charisma. If some of these numbers can not be uniquely determined, output -1 instead of such number.

Examples

stdin	stdout
3 1 0 0	0.3333333333 1.0000000000
7 1 0 1 0 1 1 0	0.6000000000 0.8333333333
2 0 1	-1 0

Problem D. Darkside

Input file:	`stdin`
Output file:	`stdout`
Time limit:	3 seconds
Memory limit:	256 megabytes

Daring duck of mystery,
Champion of right,
Swoops out of the shadows,
Darkwing owns the night.
Somewhere some villain schemes,
But his number's up.

Cloud of smoke and he appears,
Master of surprise.
Who's that cunning mind behind
That shadowy disguise?
Nobody knows for sure,
But bad guys are out of luck.
'Cause here comes
Darkwing Duck!

From the Disney Afternoon series "Darkwing Duck"

You're so tired of this guy! He is always one step ahead that makes you think that someone wrote a program for him to know all your moves in advance! You even regret a little about joining the dark side.

But now you have a plan. Even if the city of Darkwing Duck cannot be robbed, you can still work in a suburb. The suburb is build as one long street with houses represented by ASCII symbols.

It is not a Darkwing Duck zone anymore, but another defender operates there — GizmoDuck. He's a duck named Ferton dressed in a Robotic costume. Sometimes Ferton just rests at his home, and streets are patrolled by his Robotic costume only, controlled by a specially designed program.

The program is a sequence of commands '`L`' and '`R`' after which the Robot moves for the distance of one house to the left or right respectively. The Robot's mechanics don't allow him to turn twice in a row, so substrings "`LRL`" and "`RLR`" are forbidden. Additionally, no house is visited more than twice during these moves to improve the effectiveness of patrolling.

Robot lands somewhere near one of the houses, executes the sequence of commands and flies back home. He prepares a full log with labels of all houses he visited in order of their appearance (if some house was visited twice, it also appears twice in the log).

You have found a list of symbols that looks like one of these logs, and you also have a map of the suburb. You would like to know the total number of different programs (strings consisting of "`L`" and "`R`") that could possibly generate this log. Maybe it will help you somehow.

Input

In the first line of input, there is a non-empty string consisting of English letters, digits and symbols '`,`', '`!`', '`_`', '`.`' and '`-`' representing the houses in the suburb from left to right.

In the second line of input, there is a non-empty string consisting of English letters, digits and symbols '`,`', '`!`', '`_`', '`.`' and '`-`' representing the found log.

The length of each string is no more than $2 \cdot 10^5$ symbols.

Output

Output one integer: the total number of different correct programs that can lead to the found log.

Examples

stdin	stdout
Welcome_to_the_suburbia rubu	2
aa a	1

Note

In the first example, the correct programs are "LLL" and "LLR". Robot can land at position 20 ('b'), execute one of these programs and fly back home from position 17 or 19 respectively, producing the expected log.

In the second example, the only possible program is an empty string (it can be executed from any landing position).

Problem E. Edges Are Too Sharp!

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

The company "Nanozavod, Inc." is specialized on design and assembly of nanobots. Now they meet another problem on their long way. Artificial joints require pieces of nanoribbons of exact length. Developers can create nanoribbon segments of the known lengths, but they are too long. Now they are trying to cut a segment into two parts of equal length.

A nanocutter looks like a hollow polygonal prism with another prism inside it. The inner prism has very sharp edges and is used to cut nanoribbon: when a piece of nanoribbon touches any edge of the inner prism, it is immediately cut in the contact point. Unfortunately, if a ribbon touches the inner prism in more than one point but by some longer segment (for example, it contacts the entire edge of the prism at the same moment), the whole piece of nanoribbon is completely destroyed. The outer prism is used to hold a piece of nanoribbon: the endpoints of the ribbon must be put onto the side surface of the outer prism. The tension of the ribbon is exactly uniform.

So, the developers now have to find two points on the outer prism's side surface such that the segment connecting these points has exactly one common point with the inner prism, and this common point is the middle point of the segment. All edges of both prisms which connect their base faces are parallel, so the problem can reduced to the same two-dimensional problem for the base faces of the prisms. Coordinates of both prisms' vertices are integers, bottom vertices form two polygons that are strictly convex (that is, there are no 180-degree corners of polygons). The inner polygon is strictly inside the outer one (their borders have no common points). Your task is to find coordinates of ends of any segment that satisfies the conditions of the problem. It is guaranteed that at least one such segment exists.

Input

The first line of input contains two integers: M is number of outer polygon vertices and N is number of inner polygon vertices ($3 \le M, N \le 3 \cdot 10^5$).

The following $M + N$ lines contain two integers x and y each ($|x|, |y| \le 10^7$) that are coordinates of the polygon vertices (first M lines for the outer polygon, next N lines for the inner polygon). Vertices of each polygon are given in counterclockwise order.

It is guaranteed that the inner polygon is strictly inside the outer one.

Output

Print four real numbers x_1, y_1, x_2, y_2 specifying the coordinates of the nanoribbon's endpoints. The coordinates must be given with absolute error no more than $\varepsilon = 10^{-4}$ (that is, there must exist an exact solution to the problem (X_1, Y_1, X_2, Y_2) such that $|x_1 - X_1| < \varepsilon$ etc.). If there are multiple answers, print any one of them.

Examples

stdin	stdout
3 3 0 0 4 0 0 4 1 1 2 1 1 2	2.000000 0.000000 2.000000 2.000000

Problem F. Four Ways to Travel

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

A new payment system has been introduced for public transportation in city M: the electronic card "Tetrad". It is a refillable card suitable for paying for the following means of public transportation: bus, trolleybus, tram and subway.

A passenger starts using "Tetrad" by buying the card itself and putting some money on the card balance via a vending machine or a payment terminal. Then he uses his card by swiping it at a validator when boarding some means of surface transportation or entering the subway system, and some amount of money is deducted from the card balance. Note that the payment does not depend on the duration of a trip.

The common payment for using subway is 28 rubles, and 26 rubles for bus, trolleybus or tram. However, for passengers who make a lot of transfers during their travel, a new "90 minutes" plan has been developed: for 44 rubles, a passenger can travel by any number of buses, trolleybuses or trams, and also enter the subway system no more than once, during an 1.5-hour interval. The order of using different means of transportation does not matter. For example, you can take a subway train first, then take a trolleybus; or travel by bus first, then descend into the subway, take a train or two there (you swipe the card at a validator upon entering the subway system as a whole, not when changing trains), come up again and take a tram to reach the desired location; or use various means of ground transportation only. In any of the above cases, only 44 rubles will be deducted from the card for the whole trip using the "90 minutes" plan. The time when the transport arrives at a destination does not matter — only the moments when you enter a bus, a trolleybus, a tram or the subway system are essential for time management.

The electronic system determines the moment to activate the "90 minutes" plan automatically. For simplicity, let us consider ground transportation first. If there is a second trip during the 90 minute interval from the first one (inclusive), the system activates the "90 minutes" plan from the time of the first trip, and the cost of the second one is automatically decreased. Further trips which start within 1.5 hours since the first one are free of charge: you can travel without any limits.

It is getting more complicated with the subway. The "90 minutes" plan allows entering the subway system only once. In the case when, during an ongoing 1.5-hour travel interval, the passenger enters the subway system for the second time, this 1.5-hour travel interval is interrupted, and using the subway costs the usual 28 rubles. If a ground transportation takes place soon enough, the latter subway boarding will be the start of a new 90-minute travel interval.

For example, let us assume that at `10:00` you enter the subway, paying with a "Tetrad" card. The validator shows "-28" (28 rubles are deducted from your card). Then at `10:30` you exit the subway and take a tram. Only 16 rubles ($44 - 28 = 16$) are deducted from your card when boarding a tram. After that, at `10:50` you take a bus, and this time, it is legally free of charge. Then at `11:20` you enter the subway once again. 28 rubles will be deducted from card, because only one entry to the subway is included in the "90 minutes" regardless of the 1.5-hour interval from the time of the first trip. So, this second subway trip starts a brand new 1.5-hour interval. If your next trip on, say, trolleybus starts at `12:50`, then it's 16 rubles, but if it starts at `12:51`, it's 26 rubles.

This new plan is very hard to comprehend by citizens and is seen as overly complex by them. However, it would be even harder if passengers could decide when to activate the "90 minutes" plan for themselves. Currently, it is done automatically.

Let us assume that you have planned your trips for a day and have enough rubles on your "Tetrad" card balance. What amount will be deducted from the card by the end of the day? Could you pay less if you could decide when to activate the "90 minutes" plan yourself? Note that in the latter scenario, it is still not allowed to switch off or suspend the "90 minutes" plan voluntarily once it is activated. That is, once

you activate the "90 minutes" plan, you have no other choice but to pay according to the plan. You have to either wait for 1.5 hours or enter the subway system for a second time to have it wear off, and only since that moment, you once again control the time of the next activation (for example, you can activate it again instantly).

Input

The first line of input contains a single positive integer N: the number of trips. Each of the next N lines contains a description of a trip. Each description consists of the time of boarding in the format `hh:mm` (hour and minute respectively, left-padded with zeroes to two digits) and a single letter: '`S`' if this trip uses surface transportation (bus, trolleybus or tram), or '`U`' if it uses underground transportation (subway). The time of boarding and the letter are separated by a single space. The trips are described in chronological order and fit within one day (the time is ascending from `00:00` to `23:59`). Each trip lasts for at least one minute, therefore, no moment of time is given twice in the input.

Note that letter '`U`' means entering the subway system (not just switching trains while underground).

Output

Output a single line with two integers: the amount that will be deducted from the "Tetrad" card for taking these N trips, and the minimum possible amount of expenses in case you had an ability to control the activation of the "90 minutes" plan.

Examples

stdin	stdout
2 00:00 U 23:59 S	54 54
5 10:00 U 10:30 S 10:50 S 11:20 U 12:51 S	98 98
4 20:00 S 21:30 S 21:31 S 21:32 S	88 70

Problem G. Global Elephant Market

Input file:	`stdin`
Output file:	`stdout`
Time limit:	3 seconds
Memory limit:	256 megabytes

— Buy an elephant!
— Why would I want an elephant?
— Everyone asks, "What I need it for", just come and buy an elephant.
— Get off me!
— I will, but first you gotta buy an elephant!

This is an interactive problem.

Welcome to the world's largest elephant market. As you are a newbie to the market, your current goal is to learn how to analyze the current state of the market. You should be able to answer how much money you can make at any given moment, assuming that you have enough initial money to be able to make all the deals you wish. You will not perform any real trade.

The elephant market is based on an auction market model where a buyer bids a specific price for an elephant and a seller asks a specific price for an elephant. The prices are updated constantly as traders change their mind.

We assume that, given the current market situation, you can buy a few elephants but you must immediately sell them. You are trading just hand-to-mouth as you are not interested in keeping elephants even for a minute. So, you have to buy and sell the same number of elephants in total.

At the opening bell of the elephant market, no one wants to buy or sell elephants. Elephant offers are placed or withdrawn sequentially, listing what number of elephants one can sell or what number of elephants one can buy at specified prices. To keep things simple, you will only see a change of the number of elephants for buying or selling at the specified price. For example, "`buy 10 100`" means that you can now sell 10 more elephants for 100 piastres each and "`sell -3 99`" means that you can now buy 3 less elephants for 99 piastres each.

Write a program to determine the maximum possible profit (amount of money) you can make on the market after each offer to buy or sell elephants is placed. Note that as you do not actually perform any real trade, you must assume that after each new offer is placed, all the previous offers are still valid too.

Input

Your program will receive market offers sequentially in the following format. Every line of input corresponds to one offer and starts with the offer type (one of strings "`buy`", "`sell`" or "`end`") meaning an offer to buy some elephants, an offer to sell some elephants and the end of the trading session respectively. Offer "`end`" contains no additional parameters, and your program must exit immediately after it. Other offers contain two integers D and P separated by a single space as additional parameters describing a change D of the number of elephants on the market with appropriate order type and price P ($-10^6 \le D \le 10^6$, $1 \le P \le 10^9$). The total number of offers will never exceed 10^5.

It is guaranteed that both the total buy price of all elephants currently available on the market and the total sell price of all elephants currently available on the market will not exceed 2^{62}, and the total number of elephants for buying or selling with any price will never become negative.

Output

After each "`buy`" or "`sell`" offer, you must output a single integer on a separate line: the maximum

possible profit in piastres you can make after this offer is placed on the market. Do not forget to print end-of-line characters and to flush output buffers after each output.

Examples

stdin	stdout
buy 10 100 sell 4 98 buy -7 100 buy 2 99 sell 1 97 end	0 8 6 7 9

Problem H. Hanmattan

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Island Hanmattan can be represented as an $M \times N$ grid: $M+1$ streets are parallel to north-south direction, and $N+1$ avenues — to east-west direction. The distance between any two neighboring streets or avenues is equal to 1.

Naturally, crossings are identified by pairs of numbers: the street and avenue which meet on the crossing, i. e. from $(0, 0)$ to (M, N).

A taxi driver received an order to carry passenger A from one of the crossings to a railway station which is placed on another crossing. A taxi can only move along streets and avenues.

At the time of boarding, the driver got to know that some passenger B is waiting on some third crossing and wants to arrive to the same railway station. Now, the taxi driver wants to transport passenger A along some optimal (shortest) route to the station, but in addition, he wants to also take passenger B along the way. Given the coordinates of all three crossings, find out if this is possible.

Input

The first line of input contains four integers x_a, y_a, x_s, y_s: the coordinates of a crossing with passenger A and the coordinates of a crossing with the railway station. The second line contains two integers x_b and y_b: the coordinates of a crossing with passenger B. All coordinates are non-negative and do not exceed 10^6. It is guaranteed that all three crossings are pairwise distinct.

Output

Print "Yes" if it is possible to take passenger B while moving passenger A along an optimal route to the station, or "No" otherwise.

Examples

stdin	stdout
1 1 3 4 2 2	Yes
2013 1 2 2014 2014 2013	No
100 0 0 100 0 0	Yes

Problem I. Intersections

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

You are given pairwise intersections of n sets of integers. Your task is to find the initial sets or find out that it is impossible.

Input

The first line of input contains an integer n: the number of sets ($1 \le n \le 200$). It is followed by $n \cdot (n-1)/2$ lines, each of them describing one of the intersections. Each line contains three integers x, y and k ($1 \le x, y \le n$, $x \ne y$, $0 \le k \le 10$) followed by k integers a_1, a_2, ..., a_k ($1 \le a_i \le 10^4$). Here, x and y are indices of the intersected sets, k is the cardinality of intersection and a_i are the elements of intersection. It is guaranteed that all elements of intersections are distinct, and for each pair of sets, their intersection will be described exactly once.

Output

If there are no such sets, the only line of output must contain a single word "No". Otherwise, the first line must contain a single word "Yes". The next n lines must contain the descriptions of such sets in the following format. The first number c in $(i+1)$-st line of output must be the cardinality of i-th set. It must be followed by c pairwise distinct integers: the elements of i-th set. The total number of elements in all sets must not be greater than 10^6.

Examples

stdin	stdout
3 1 2 2 1 2 1 3 2 3 4 3 2 0	Yes 4 1 2 3 4 2 1 2 2 3 4
3 1 2 2 1 2 1 3 2 3 2 3 2 1 3	No

Problem J. Jigsaw Puzzle

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

At an online course in discrete mathematics, one of the recent lectures was devoted to correct domino coverings of different rectangular fields of size $n \times m$ consisting of square cells. Some cells may be cut from these fields. The professor called such fields *jigsaw puzzles*. To solve a jigsaw puzzle means to find a covering of the given field with dominoes of size 1×2 such that the following conditions hold:

- the dominoes cover all the non-cut cells on the field;
- all the dominoes fully lie inside the field;
- the dominoes do not intersect each other;
- the dominoes do not lie on any of the cut cells.

Now the professor wants to know how understandable his lecture was. To find this out, he decides to give jigsaw puzzles to all students to solve. The professor doesn't want to make students' lives too difficult, so he decided that each jigsaw puzzle will have at least one solution. But he also wants everybody to do their homework themselves, and therefore, he wants all jigsaw puzzles to be different.

Before the professor will create his jigsaw puzzles, he wants to know how many different puzzles exist. Can you help him?

Your task is to compute the number of different fields of size $n \times m$ that have at least one correct domino covering. Remember that some cells of a field could be cut. This number can be quite large, so you must compute it modulo $10^9 + 7$.

Two fields are considered different if there exists a cell on an $n \times m$ grid that is cut from one field but not cut from the other. A field can not be reflected or rotated.

Input

The input contains only one line with two integers n and m ($1 \le n \le 6$, $1 \le m \le 500$) separated by a single space.

Output

The output must contain a single line with a single integer: the answer to the problem.

Examples

stdin	stdout
1 2	2
2 3	18

Problem K. Top K Elements

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	8 megabytes

Alexey is attending the "Algorithms and Data Structures" course at his university. He has already learned quite a few interesting algorithms and has recently got a new task for his homework.

Given a large array of n elements, he has to find out the k largest of them.

"Sounds very simple!" — Alexey wondered and immediately started working on it.

5 minutes, 10, 15... and at last, the algorithm is written.

Tests are run...

"Oh... Memory limit exceeded!" — Alexey started to lose his temper. But no matter how hard he tried, he always got a "memory limit" error.

Suddenly a great idea came to his mind.

"ACM!" — he exclaimed. He remembered that today is a very important day when the most gifted people from all over the Moscow subregion participate in the quarterfinal programming contest.

As far as Alexey knows, programmers are not only brilliant but also very kind. He has decided to ask you to help him make his homework.

Note that your algorithm has to be very fast and VERY LIMITED IN MEMORY!

Input

The first line of input contains two integers n and k ($1 \le n \le 10^8$, $1 \le k \le 2 \cdot 10^5$, $k \le n$) separated by a single space. The following line contains five integers x_{-1}, x_0, A, B and C separated by single spaces, each of them between 0 and $2^{31} - 1$ inclusive.

The array consists of numbers x_1, x_2, ..., x_n where $x_i = (A \cdot x_{i-2} + B \cdot x_{i-1} + C) \bmod 2^{31}$.

Output

You must output the largest k elements of the array in non-increasing order. The elements must be separated by single spaces.

Examples

stdin	stdout
5 3 0 0 0 1 1	5 4 3

Note

In the example, the generated sequence is 1, 2, 3, 4, 5.

Problem L. Liar, Liar!

Input file:	`stdin`
Output file:	`stdout`
Time limit:	3 seconds
Memory limit:	512 megabytes

You and your friends have very boring Sociology studies. To make it even worse, your lecturer sometimes invites some famous science freak or even a politician. You became so tired of this non-stopping flood of lies and wrong logic that you have finally invented a funny game named "Liar, Liar!".

You are preparing a list of examples from real life in advance. Each of them consists of a set of properties. For each of those properties, it is known whether it is true or false in this particular example.

An example is said to show a contradiction in a speech if there is a set of implications from speech that subsequently implies a false example property from a true example property.

When some example contradicts with the speech of "another extraordinary professor" for the first time, you stand up and shout "Liar, liar!" telling the number of that example.

You want to prepare well for the upcoming lecture. You have found a video of the previous lecture of this man (which was exactly the same) and have prepared a list of examples. Now you want to compute in advance when you should stand up and shout.

The lecture consists of the number of implications "property A implies property B", but science freaks (and of course politicians) are very sneaky people and they never imply one property twice from other properties to make their lies harder to find. So, no two distinct statements of the forms "A implies B" and "C implies B" for any (not necessarily distinct) properties A, B and C can ever be found in one speech.

Input

The first line of input contains an integer M: the number of statements in the speech ($0 < M \le 500\,000$). The next M lines contain descriptions of implications in the order of their appearance in the speech, one implication per line. Each implication is represented by two integers a_i and b_i ($a_i \ne b_i$, $0 < a_i, b_i \le 500\,000$). This means that property a_i implies property b_i. All b_i are distinct.

The next line there contains an integer Q: the number of prepared examples ($1 \le Q \le 500\,000$). The following Q lines contain descriptions of these examples, one per line. The first number in each such description, t_i, is the number of properties which are true for the given example. Then t_i integers are given: the numbers of these properties. After that comes f_i, the number of properties which are false for the given example, and then f_i integers: the numbers of these properties. All numbers of properties on any single line are pairwise distinct, greater than 0 and do not exceed 500 000. Additionally, it is guaranteed that, for each line i, $t_i + f_i > 0$. The sum of all t_i and f_i over all i does not exceed 500 000.

Output

Output Q lines: for each example, output the number of the first statement of speech after which the example becomes self-contradictory. The statements of the speech are numbered from 1. If an example does not become self-contradictory after all M statements of the speech, output -1 for that example.

Examples

stdin	stdout
6 1 2 2 5 5 7 5 6 2 3 2 4 3 2 1 5 2 3 7 1 2 1 6 1 6 1 2	3 4 -1
5 1 2 2 3 3 4 4 5 5 1 6 1 2 1 4 1 4 1 2 1 2 1 3 1 3 1 2 1 1 1 5 1 5 1 1	3 5 2 5 4 5