Problem A. Rectangle Query

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	768 mebibytes

This is an interactive problem.

You are given n points on the plane with integer coordinates. You are to answer the following queries: "Consider all points inside or on the border of a rectangle given by $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$. How many different x-coordinates and different y-coordinates are there among the chosen points?"

Note that this problem is interactive and you must answer all previous queries before the given one.

Input

The first line contains single integer n, the number of given points ($1 \le n \le 50\,000$).

The next n lines contain coordinates of the points x_i and y_i, one point per line ($0 \le x_i, y_i \le 500\,000$). It is guaranteed that all given points are different.

The following line contains the number of queries q ($1 \le q \le 50\,000$).

Each of the next q lines contains four integers x_1, y_1, x_2, y_2: coordinates of the opposite corners of the rectangle ($0 \le x_1 < x_2 \le 500\,000$, $0 \le y_1 < y_2 \le 500\,000$). Note that each query i ($i > 1$) will be available only after you answered query $(i - 1)$.

Output

Output q lines, one for each query. Each line must contain two integers separated by space: the numbers of different x and y coordinates respectively. Don't forget to flush the output after printing each answer.

Example

standard input	standard output
4 2 0 2 1 1 1 1 2 4 0 0 2 2 1 1 2 2 1 0 2 1 0 0 1 1	2 3 2 2 2 2 1 1

Problem B. Game With A Fairy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

This is an interactive problem.

You met a fairy in a beautiful forest glade. The fairy is in good mood and would like to give you a present, but not before you best her in a game though.

There are n trunks in the glade; the trunks are numbered from 1 to n. Some of the trunks (**possibly all, and at least one**) contain magical treasure. You can choose several trunks (possibly all) and list their numbers to the fairy. If none of them contain treasure, then you are unlucky and do not get any treasure. However, if more than one of the chosen trunks contains treasure, the fairy thinks you are too greedy and does not give you any treasure either. If **exactly one** of the trunks chosen by you contains treasure, then you get the treasure and are expelled from the forest (that is, the game ends).

You can make at most **200** guesses before the night falls and the fairy becomes annoyed with you. It is guaranteed that the fairy is honest and does not move any treasure after you enter the forest. Can you best the fairy and get the precious treasure?

Interaction Protocol

At the start of the interaction, a single integer n is fed to your program on a separate line. In all test cases except the first one, $n = 10^4$. The first test case coincides with the sample. In each test, the trunks which contain treasure are fixed in advance and stay the same each time a solution is evaluated.

For each query you make, print a single line of n characters; i-th of the characters must be "`1`" if the i-th trunk is chosen for this query, or "`0`" if the i-th trunk is not chosen.

After each query, your program is fed with a line containing "`+`" if you won the treasure, or "`-`" if you didn't win the treasure (note that in this case, you are not told whether no trunks contain treasure, or there are multiple trunks with treasure). After receiving the "`+`" answer, your program must terminate immediately.

Do not forget to end your lines with "new line" characters, and flush your output after each query.

Example

standard input	standard output
3 - - +	100 111 010

Note

In the sample test, magical treasure is in trunks 2 and 3.

Problem C. Portkeys

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

There are n portkeys located on a straight line. Each portkey has three characteristics: x_i (coordinate of the portkey), l_i and r_i. A portkey can be used to teleport to different points on the line. That is, if you are standing at the point x_i, you can use the portkey to instantly teleport to any point y that satisfies $l \le |x_i - y| \le r$. You are not allowed to move along the line without using a portkey. Initially, you are located at the point x_1 (the location of the first portkey).

You are also given m target points on the line. For each target point, find the smallest number of portkeys to be used to get to this point from the start.

Input

The first line contains one integer n, the number of portkeys ($1 \le n \le 2 \cdot 10^5$).

Each of the next n lines contains three space-separated integers x_i, l_i, r_i: the characteristics of the i-th portkey ($-10^9 \le x_i \le 10^9$, $0 \le l_i \le r_i \le 10^9$).

The next line contains one integer m, the number of target points ($1 \le m \le 2 \cdot 10^5$).

The last line contains m space-separated integers y_i: the coordinates of the target points ($-10^9 \le y_i \le 10^9$).

Note that an arbitrary number of portkeys and target points can be located at the same point on the line.

Output

On the first line, print space-separated answers for all target points in the same order the points are given in the input. The answer for a target point is the smallest number of portkeys that are needed to be used in order to get to the point, or -1 if the target point is unreachable via portkeys.

Examples

standard input	standard output
1 1 2 3 4 1 2 3 4	0 -1 1 1
2 0 3 5 5 6 7 5 3 3 12 5 6	1 1 2 1 -1

Problem D. Maximal Common Subpair

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	768 mebibytes

A pair of strings (x, y) is called a subpair of string s if there are strings w, v and u (possibly empty) such that $s = wxvyu$. Strings x and y can be empty as well.

A pair of strings (x, y) is called a common subpair of strings s_1 and s_2 if it is a subpair of both s_1 and s_2.

The length of a subpair (x, y) is defined as $|x| + |y|$, that is, the sum of lengths of x and y.

You are given two strings s_1 and s_2 consisting of lowercase Latin letters. Find their common substring pair with maximal possible length.

Input

The first line contains the string s_1.

The second line contains the string s_2.

Both strings are non-empty, consist of lowercase Latin letters and contain no more than $2 \cdot 10^5$ letters each.

Output

The first and the second line of the output must contain the first and the second strings of the chosen subpair, respectively. Any of these strings can be empty as well. If there are several common subpairs with maximal length, you can output any of them.

Examples

standard input	standard output
abacaba cabina	cab a
abba acdc	 a
empty ans	

Note

In the first test, the common subpair with maximal length is ("cab", "a").

In the second test, there are two common subpairs with maximal length: ("", "a") and ("a", "").

In the third test, the strings contain no common substrings, thus the maximal length is 0, and the answer contains two empty strings.

Problem E. k-transpositions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	256 mebibytes

Given the identity permutation of n elements, you need to find the number of permutations one can obtain from it using no more than k transpositions. Since the desired number can be rather large, output it modulo $10^9 + 7$.

A *transposition* is an operation that swaps two different entries of the permutation.

Input

The only line of the input contains two space-separated integers n and k ($1 \le n \le 10^9$, $0 \le k \le 3000$).

Output

Output one integer: the answer to the problem modulo $10^9 + 7$.

Examples

standard input	standard output
4 1	7
4 2	18
7 7	5040

Problem F. PQ tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	256 mebibytes

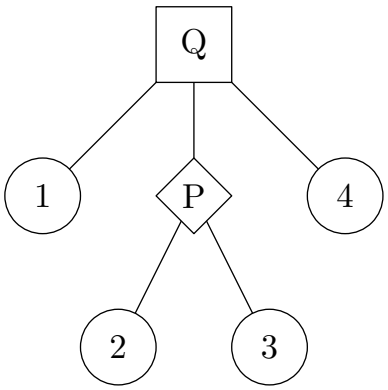
A *PQ tree* is a data structure that represents a family of permutations. The structure is utilized for solving various computational problems such as testing graph planarity or recognizing interval graphs.

A PQ tree on n elements is a rooted tree with vertices of three types: a *P-vertex*, a *Q-vertex*, or a *leaf*. Any P-vertex has at least two children (and its children are initially *unordered*), any Q-vertex has at least three children (and its children are initially *ordered*), and any leaf has no leaves. There are exactly n leaves, and they are associated with distinct integer values from $\{1, \ldots, n\}$.

A permutation σ of numbers $\{1, \ldots, n\}$ is *represented* by the given PQ tree if it possible to order the children of all internal (P and Q) vertices in the following way:

- children of a P-vertex are ordered arbitrarily;
- the order of children of a Q-vertex is either left unchanged or *reversed*;
- when the tree is traversed starting from the root according to the chosen children ordering, the leaves are visited in the order given by the permutation σ.

For example, consider the following PQ tree:



Evidently, the tree represents permutations $(1, 2, 3, 4)$ and $(4, 3, 2, 1)$, but does not represent permutations $(2, 1, 4, 3)$ (since in every ordering, the numbers 2 and 3 must be adjacent) and $(1, 4, 3, 2)$ (since 4 must be located either at the beginning or at the end).

We will consider two PQ trees identical if every permutation represented by the first tree is represented by the second tree, and vice versa; otherwise, the trees are distinct (note that this means that reversing the order of children of a Q-vertex does not change the PQ tree).

You are given k permutations of numbers $\{1, \ldots, n\}$. Count the number of distinct PQ trees on n leaves which represent all given permutations. Print the answer modulo $10^9 + 7$.

Input

The first line contains two integers k and n ($1 \le k, n \le 500$).

The next k lines contain n space-separated integers each: the given permutations. On each line, the numbers are pairwise distinct and belong to the set $\{1, \ldots, n\}$.

Output

Print the answer to the problem modulo $10^9 + 7$.

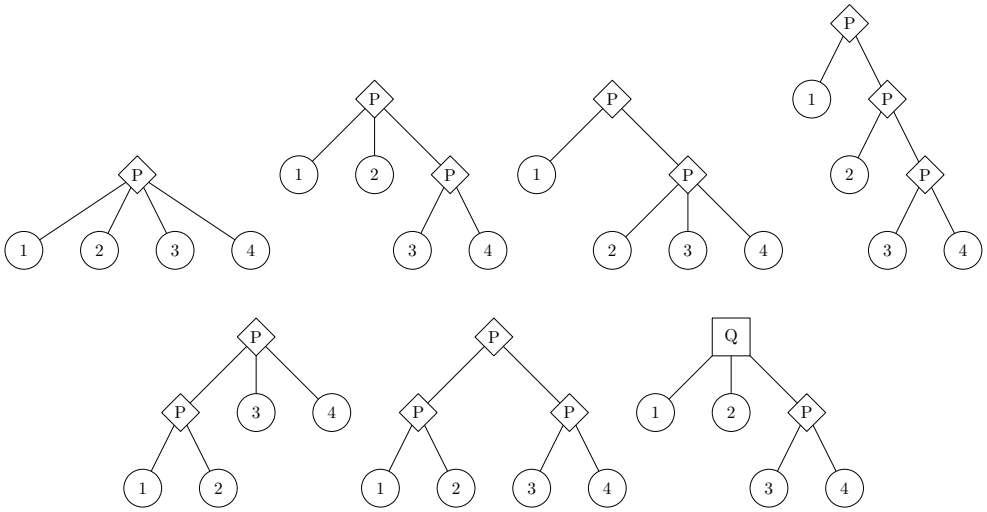
Examples

standard input	standard output
1 2 1 2	1
2 4 1 2 3 4 1 2 4 3	7
1 5 1 2 3 4 5	82

Note

The only tree on two elements is the P-rooted tree with two leaf children (note that Q-vertex must have at least three children).

The seven trees satisfying the second sample are pictured below:



Problem G. Random Walking

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

In Berland, there are n cities and $n - 1$ distinct bidirectional roads connecting the cities. The cities are numbered from 1 to n, and there is a way along the roads between each pair of cities (so, Berland can be considered a tree).

When a tourist flies from other country to the city u and wants to visit the city v, he repeats the following procedure until he reaches the city v: the tourist chooses a *neighbor city* (that is, the city which is connected by a road with the city our tourist is located in) and moves to it. Traveling every road between adjacent cities takes exactly one hour. As Berland has no signposts, the tourist does not know where to go, so each time he selects a neighbor, the selection is random, and all neighbors are equally probable. Note that the tourist does not remember the roads he traveled before, and thus can choose the road he used to arrive in the city as well as every other road.

The king of Berland is concerned about the average time spent by a tourist, so he wants to know how long it takes (on average) to reach v_i starting in u_i for some pairs u_i and v_i. Answer his questions.

Input

The first line contains one integer n, the number of cities in Berland ($1 \le n \le 10^5$).

Next $n - 1$ lines contain descriptions of the roads as pairs of integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) meaning that there is a road between a_i and b_i. It is guaranteed that the given graph is a tree.

The following line contains the only integer q, the number of questions ($1 \le q \le 2 \cdot 10^5$).

Next q lines describe the king's questions: i-th of these lines contains two space-separated integers u_i and v_i ($1 \le u_i, v_i \le n$).

Output

For each question, print the average time (in hours) a tourist will spend to get from u_i to v_i. The answer is considered to be correct if its absolute or relative error does not exceed 10^{-9}.

Example

standard input	standard output
3 1 2 2 3 3 1 2 2 1 3 3	1.000000 3.000000 0.000000

Problem H. Sasha And Swag Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	768 mebibytes

Hey you, swagger and [a bad person]! You [bothered] everybody. Shove your suffix tree in your [censored].

Well-wisher

Sasha enjoys strings and problems on string theory. Especially he loves suffix structures. As Sasha loves **TopForces** community he is going to write an entry named *"On suffix trees, part 37"*. Surely, he wants everybody to understand what the article says, so he decided to decorate it with some nice examples. After a recent contest on **TopForces**, Sasha achieved the rank "International swagger", and now he wants to match it. To achieve this, Sasha wants to choose the swaggest strings among some given set.

Of course, Sasha's concept of string swagness is based on suffix structures. To be precise, on suffix trees. To calculate the swagness of a string, Sasha performs the following actions:

1. He constructs the compressed suffix tree of the string.

   Let us recall that a suffix tree is the trie built from all suffixes of the string. A compressed suffix tree is the same trie in which consecutive edges are merged. See the notes section for an example.

2. On each edge of the compressed suffix tree, he counts the number of distinct non-empty substrings of the string which is written on the edge.

The swagness of the string is defined as the sum of the calculated values among all edges. Unfortunately, Sasha is not so good with the suffix trees as he wants you to think, and can't handle the problem on his own. Help him!

Input

The first line contains an integer n ($1 \le n \le 10^5$).

Each of the next n lines contains a string. The i-th line contains a non-empty string s_i. Each string consists entirely of lowercase Latin letters. The total length of all strings s_i does not exceed $5 \cdot 10^5$.

Output

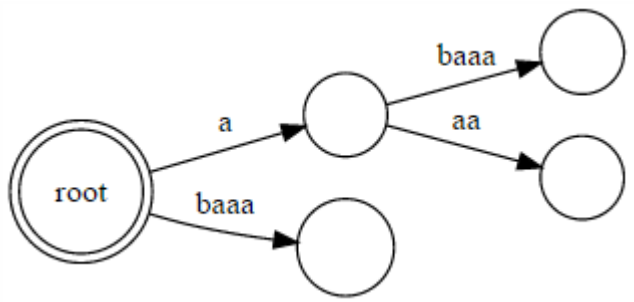
Print n integers, one per line. The i-th number must be the swagness of the string s_i.

Examples

standard input	standard output
3 umqra umnik merkurev	35 35 101
1 abaaa	17

Note

Compressed suffix tree for string *abaaa*:



Edges are a, aa, $baaa$ and $baaa$. They have 1, 2, 7 and 7 distinct non-empty substrings, respectively. So, the answer is $1 + 2 + 7 + 7 = 17$.

Problem I. Tree Confrontation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

In the ancient times, the country of Barelia had n cities connected by bidirectional roads. Each pair of cities had exactly one simple path between them (that is, the country topology was a tree).

For the longest period of time, there were two opposing wizard factions living in Barelia: white mages and black mages. At every moment of time, each Barelian city could have been occupied by either a white mage or a black mage (it was possible that a city was free from mages of any kind, but no two mages could occupy the same city, even if they belonged to the same faction).

It is known that at all times, exactly a of the cities were occupied by white mages, and exactly b of the cities were occupied by black mages. Also, for each pair of cities occupied by the mages of the same faction, there must have been a path between them such that every city on the path was occupied by the same faction (that is, at all times, the cities occupied by a certain faction formed a connected subtree).

From time to time one of the mages could decide to travel to another city. Suppose that a mage was located at the city v. To travel, he chose a path $P_1, \ldots, P_k$ such that $P_1 = v$, all the cities $P_1, P_2, \ldots, P_{k-1}$ are occupied by the mages of the same faction, and the city P_k is free. After that, the mage moved from the city P_1 to the city P_k (that is, the city P_1 became free, and the city P_k became occupied by a mage of the same faction). Note that after the travel, the cities occupied by the same faction still had to form a connected subtree.

It is historically important to know if some conjectures about the mages' locations contradict. Consider two configurations of mages' locations A and B; we will say that A *does not contradict with* B if A can be transformed to B via a sequence of valid mages' travels. Clearly, the relation "A does not contradict with B" is an equivalence relation; thus all possible configurations can be divided into maximal mutual non-contradictory classes.

In order to estimate the research expenses, the Barelian Historian Society asked you to determine the number of non-contradictory configuration classes.

Input

The first line contains three space-separated integers n, a and b ($1 \le n \le 200\,000$, $1 \le a, b \le n$).

The next $n - 1$ lines contain description of the roads. The i-th line contains two space-separated integers u_i and v_i: the numbers of the cities connected by the i-th road ($1 \le u_i, v_i \le n$). It is guaranteed that the given roads form a tree.

Output

Print the number of non-contradictory configuration classes. Note that, if there are no valid configurations, the answer is 0.

Examples

standard input	standard output
4 1 1 1 2 2 3 3 4	2
5 1 1 1 2 1 3 1 4 1 5	1
5 2 1 1 2 1 3 1 4 1 5	4
5 2 2 1 2 1 3 1 4 1 5	0

Problem J. Two Airlines

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

This is an interactive problem.

In Berland, there are two airlines: Aeroshipping and Wicktory (it was called Kindplane earlier). Since Berland is a developed country, for each pair of cities, there is an airline which provides flights between them. Since Berland is an economical country, for each pair of cities, there is only one such airline.

The cities of Berland are numbered from 1 to n (so there are n cities in Berland). One tourist wants to travel around Berland, that is, he wants to visit each city exactly once and finish his trip in the city where he started. The starting city can be chosen arbitrarily.

As the tourist doesn't want to annoy the airlines' staff, he wants to change airline no more than once. Unfortunately, the tourist does not know which airline controls each of the possible flights. To learn that, he can ask questions of the form "which airline controls the flight from city u to city v"? Obviously, the tourist doesn't want to waste too much time, so he decided to ask no more than $2n$ such questions.

Help the tourist to make such a route! It is guaranteed that the answer exists.

Interaction Protocol

At first, you are given only the integer n, the number of cities in Berland ($3 \le n \le 777$).

Each time you want to know which airline connects cities u and v ($1 \le u, v \le n$; $u \ne v$), print a single line "`?` u v".

After each query, you will be given a line with one character which will be either "`A`" or "`W`": the first letter of the name of airline which connects the respective pair of cities.

If you make more than $2n$ queries, your solution will be terminated with outcome "`Wrong Answer`".

When you are ready to offer a route, print a single line "`!` a_1 a_2 ... a_n" which lists the numbers of cities in your trip in the order of visiting them. In this trip, all flights between adjacent cities must be provided by the same airline, or there must exist an integer k such that each flight between adjacent cities among $a_1, \ldots, a_k$ is provided by one airline, and each flight between adjacent cities among $a_k, \ldots, a_n, a_1$ is provided by the other airline (remember that the tourist must travel from a_n to a_1 at the end of his trip!). It is guaranteed that there exists at least one solution satisfying the above constraints.

After printing a route, your program must exit immediately.

Do not forget to end your lines with "new line" characters, and flush your output after each query.

Example

standard input	standard output
5	? 1 5
A	? 5 4
A	? 4 3
W	? 3 2
W	? 2 1
W	! 1 5 4 3 2