Problem A. Garland Checking

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

When preparing to New Year Eve, Queen Amidala always checks her electric garland to ensure it works. The garland consist of n lamps which are numbered from 1 to n; there are $(n-1)$ pairs of them which are allowed to be connected with wires to form a special structure. Each wire in the structure connects exactly two different lamps, and all the lamps are connected together with the wires.

However, the garland is so huge that this year, the Queen decided not to build the whole structure at once. However, she still needs to check each wire. So, she will connect some of the wires, then check if electric current can pass from some lamps to some others, then disconnect some of the connected wires, connect other ones and so on. More formally, the Queen will perform a sequence of operations. Each operation will be one of the following:

- Connect two different lamps with a wire;
- Remove a wire that connects two lamps;
- Test if electric current can pass between two lamps by the wires that are currently connected.

In order to properly test the garland, she will always follow the garland structure. This means that she will only connect pairs of lamps which are allowed to be connected. Moreover, Amidala will connect and disconnect any of the $(n-1)$ allowed pairs exactly once.

The Queen was almost starting to check the garland, but she realized that not all her tests will be meaningful because some pairs of lamps will be disconnected at the time she checks them, and the current will not pass even if the garland is in fact working. She asked Anakin Skywalker to help in checking her plan of checking the garland. Anakin is sure that it's boring to proceed all the operations manually, so he asked you to write a program which will process all three types of operations described above; for each test operation, the program must answer whether the two lamps are connected by wires of not. Anakin shouldn't disgrace himself in Amidala's eyes, so don't let him down!

Interaction Protocol

At start, your program will be given an integer n ($1 \le n \le 100\,000$). After that, you have to process queries one by one. Each query will be entered on a separate line in one of the following forms:

- "`C` a b", where a and b are integers ($1 \le a, b \le n$, $a \ne b$). This query means that the Queen is connecting lamps a and b with a wire. There will be exactly $(n-1)$ such queries.
- "`D` a b", where a and b are integers ($1 \le a, b \le n$, $a \ne b$). This query means that the Queen is disconnecting lamps a and b. It's guaranteed that at the moment this query is entered, lamps a and b are connected by a wire; note that pairs (a, b) and (b, a) correspond to the same wire. There will be exactly $(n-1)$ such queries.
- "`T` a b", where a and b are integers ($1 \le a, b \le n$). This query means that the Queen is testing if electric current can pass between lamps a and b. You must print a line containing "`YES`" (without quotes) if lamps a and b are reachable from each other by wires, and "`NO`" otherwise. Do not forget to flush your output.
- "`E`", this means the end of checking, your program must immediately terminate after receiving this query.

There will be no more than 300 000 queries.

It is guaranteed that the queries will always follow the garland's structure, and each possible wire will be connected and disconnected exactly once.

Initially, all possible wires are disconnected.

Example

standard input	standard output
3 C 1 2 C 2 3 T 1 2 D 2 3 D 1 2 T 1 2 E	 YES NO

Problem B. Desert Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

Tatooine... Sand people, Jawas and, of course, thousands of bounty hunters. And another fragment of Star Map. This adventure promises to be quite interesting!

In the quest of a fragment our heroes go to the Sand people's village. It consists of n houses and is so small so that there is exactly one path between any two houses. Houses are numbered from 0 to $n-1$ and have number plates with unique signs. Initially any house u has sign u on its number plate. The house of chief initially has sign 0.

Strange things happen in this village, and this is the next puzzle for our heroes.

Sometimes unknown person starts his way from the house with sign u and goes to the house with sign v. Every time he passed house (including houses with signs u and v), he took off the number plate of this house and put it on the top of pile of number plates in his bag (initially his bag is empty). But then he goes back and passes house without number plate, he takes off the sign from the bottom of his bag and attaches the number plate to this house. As a result, sequence of signs on his path is reversed. For example, if the sequence of the signs on the path from the house with sign 1 to the house with sign 2 is $1 \to 3 \to 0 \to 2$, then after actions of unknown person it will be $2 \to 0 \to 3 \to 1$.

Chief of the village often wants to know the sign of the house that belongs to path from a house with sign u to the house with sign v and is closest to his house. Since the signs are often shuffled, it is an impossible problem to him.

Our heroes should help chief and answer all his questions. Maybe it will help them in their searches?

Input

First line of the input contains one integer n ($1 \le n \le 10^5$) — number of houses in the village.

Each of the next $n-1$ lines contains two integers u and v ($0 \le u, v \le n-1$, $u \ne v$) — signs of the connected houses.

Next line contains one integer q ($1 \le q \le 10^5$) — numbers of events, followed by q lines.

Each of next q lines contains three integers t, u and v ($1 \le t \le 2$, $0 \le u, v \le n-1$).

Here, $t = 1$ means the strange unknown person goes from the house with sign u to the house with sign v and reverses the sequence of signs of his path.

And if $t = 2$, you should answer the chief's question and output the sign on the such house w that belongs to the path from a house with sign u to the house with sign v and is closest to the chief's house.

Output

For all events of the second type ($t = 2$) output one integer — the answer of the chief's question. Each answer should be written on a separate line.

Examples

standard input	standard output
8 0 1 0 2 0 3 1 4 1 7 2 5 3 6 4 1 4 5 2 5 7 1 4 6 2 4 6	2 0

Problem C. Galilei

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

Galileo Galilei was a famous astronomer. He was the first to use the telescope to look at the sky. He discovered many celestial bodies, and it appeared that tracking all of them in mind was quite hard... There is a rumor that he invented a tracking system for his inventions and discoveries called *Galilei's inventions tracker*, or simply *git*. It allowed to save many versions of various information. For example, if you would like to look at some parameters of the star you observed a few months ago, you could just query this system. Your task is to reproduce the behavior of a simplified version of this tracker. A brief description follows.

A *commit* is a representation of the entire system state. Commits are named with sequential integers starting from 0. Initially, there is only one commit number 0. Commits are ordered as a rooted tree with root at 0: every commit except the root has exactly one parent. Commit a is called an *ancestor* of b if it is possible to come from b to a going up to parent several times. One commit is always selected as the *working copy*. Several operations are supported:

- `co` x, "checkout": select commit x as the working copy.
- `ci`, "commit": create a new commit whose parent is the current working copy and select it as the working copy. The new commit gets the lowest possible id.
- `re` x, "rebase": let your working copy be y. Find a commit t which is the lowest common ancestor of x and y. Copy the path from t to y (not including t) on top of x. New commits are numbered as follows: at first, the closest one to x gets the lowest possible number, then the second commit gets the next lowest possible number, and so on. The last copied commit becomes your working copy.

  If x is an ancestor of y, nothing happens and the path is not copied. If y is an ancestor of x, then the path is not copied and x becomes the working copy.

Your task is to implement all these operations.

Input

The first line of input contains one integer n ($0 \leqslant n \leqslant 10^5$). Each of the next n lines contains one of the following commands:

- `co x`: checkout at commit x
- `ci`: commit
- `re x`: rebase over commit x

It is guaranteed that the commit number is always valid. Additionally, it is guaranteed than after all operations, there will be no more than 10^{18} commits.

Output

After every rebase operation, print the number of the working copy commit after this operation.

Example

standard input	standard output
7 ci ci co 0 ci ci re 1 re 4	6 9

Problem D. For Fun and Lulz

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Together with a group of like-minded persons, you are investigating and inventing new awesome ways to spend free time. Sport games? Done. Board games? Done. ACM? No, thanks, we mean really free time. Random graphs? Yes, that is a freaking interesting one!

The process is simple. Initially, you get an empty graph: there are N nodes and no edges in it. On each step, a random edge is added to the graph. A random edge is formed in the following way:

- A random node v_i is chosen (each node can be chosen equiprobably).
- A random node u_i is chosen (each node can be chosen equiprobably, **including** v_i).
- A random integer w_i from the range $[1, 10^9]$ is chosen (each value can be chosen equiprobably).

It is guaranteed that the input data, except the example from the problem statement, was generated according to the way described above. The generation process used the quality of equiprobability that is more than enough to accept this task (actually, it was a pseudo-random linear congruential generator, as we see no reason to use Mersenne twister or something else in this problem).

Your goal is to determine the weight of the minimal spanning forest right after each new edge appears. Why? Because you can!

Input

The first line of input contains two integer numbers N and M ($1 \le N \le 100\,000$, $1 \le M \le 200\,000$). The next M lines describe the random edges in the order of their addition. Each description consists of three integers v_i, u_i and w_i ($1 \le v_i, u_i \le N$, $1 \le w_i \le 10^9$).

Output

Print M lines, each containing one integer number: the weight of the minimal spanning forest after adding the i-th edge.

Example

standard input	standard output
5 8 1 2 4 1 1 1 2 3 3 4 5 2 5 1 4 1 3 1 1 3 2 2 5 3	4 4 7 9 13 10 10 9

Note

The example above is not generated randomly, but all other tests are generated using the method described.

Problem E. Link Cut Digraph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

After reading the paper *Incremental Topological Ordering and Strong Component Maintenance*, you came up with the following problem.

You are given a graph with n vertices. There are no edges initially. There are m operations. Each operation is first to add a given directed edge to the graph, and then to output the number of pairs (u, v) $(1 \leq u < v \leq n)$ such that u is reachable from v and v is reachable from u.

Can you implement the algorithm described in the paper in an ICPC contest?

Input

The first line contains two integers n and m ($1 \leq n \leq 10^5$, $1 \leq m \leq 2.5 \cdot 10^5$).

Each of the following m lines contains two integers u and v ($1 \leq u, v \leq n$) indicating a newly added directed edge. Parallel edges and self-loops are allowed.

Output

Output m integers, one per line: the requested number of pairs after adding each given edge.

Example

standard input	standard output
4 6 1 2 2 3 2 1 3 4 4 3 3 2	0 0 1 1 2 6