E. Filthy Rich Trees

time limit per test: 4 seconds
memory limit per test: 256 megabytes

You may have heard the saying "money doesn't grow on trees." Obviously, whoever came up with it stunk at both botany and programming. You've learned that creating trees which grow obscene amounts of money is so trivial, that it's left as an exercise to the reader. Time for the real challenge: profiting off your invention!

You have rooted your money-tree at node 1, and each node has a specified moola value. The amount of money that some subtree rooted at node x will produce is equal to the product of the moola values of all nodes in x's subtree. Initially, all nodes will have a moola value of 1.

Now, q times, one of two things happen, given to you in the form of "t x y":

If $t = 1$, you give node x a special fertilizer which sets its moola value to y.

If $t = 2$, a customer comes in and asks "how many times more valuable is subtree x than subtree y"? Formally, what is the value of subtree x divided by the value of subtree y. Also, you don't like printing massive numbers, so subtree x is more than 10^9 times greater than subtree y, only print "1000000000".

Input

The first line will contain a single integer n, the number of nodes in the tree. $n - 1$ lines follow, each containing two different integers, describing the edges of the tree. Additional constraint on input: these edges will form a tree.

After that there will be a single integer q: the number of queries. q line follow, each of the form txy, as described above.

$1 \leq n, q \leq 3 * 10^5$

$1 \leq t \leq 2$

$1 \leq x, y \leq n$

Output

For each query of type 2, print a single line with one real number: how many times more valuable subtree x is than subtree y. HOWEVER, if this value is >= 10^9, then print just 1000000000 instead.

Your answer will be accepted if your answers to all queries have absolute or relative error of at most 10^{-6}.

Examples

input Copy

```
1
1
1 1 1
```

output Copy

input Copy

```
3
3 2
2 1
3
2 2 2
2 1 2
2 3 3
```

output Copy

```
1.0000000000
1.0000000000
1.0000000000
```

input Copy

```
5
4 2
1 4
5 4
3 4
5
2 5 2
1 5 4
1 5 5
1 5 4
2 5 4
```

output Copy

```
1.0000000000
1.0000000000
```