War

Time limit: 1.0s **Memory limit:** 256M

A war is being lead between two countries, A and B. As a loyal citizen of C, you decide to help your country's espionage by attending the peace-talks taking place these days (incognito, of course). There are n people at the talks (not including you), but you do not know which person belongs to which country. You can see people talking to each other, and through observing their behaviour during their occasional one-to-one conversations, you can guess if they are friends or enemies. In fact what your country would need to know is whether certain pairs of people are from the same country, or they are enemies. You may receive such questions from Cs government even during the peace-talks, and you have to give replies on the basis of your observations so far. Fortunately, nobody talks to you, as nobody pays attention to your humble appearance.

Now, more formally, consider a black box with the following operations:

`setFriends(x,y)` shows that x and y are from the same country

`setEnemies(x,y)` shows that x and y are from different countries

`areFriends(x,y)` returns `1` if you are sure that x and y are friends and `0` otherwise

`areEnemies(x,y)` returns `1` if you are sure that x and y are enemies and `0` otherwise

The first two operations should signal an error if they contradict with your former knowledge. The two relations `friends` (denoted by `~`) and `enemies` (denoted by `*`) have the following properties:

`~` is an equivalence relation, i.e.:

1. If $x \sim y$ and $y \sim z$, then $x \sim z$ (The friends of my friends are my friends as well.)
2. If $x \sim y$, then $y \sim x$ (Friendship is mutual.)
3. $x \sim x$ (Everyone is a friend of himself.)

`*` is symmetric and irreflexive:

1. If $x * y$, then $y * x$ (Hatred is mutual.)
2. Not $x * x$ (Nobody is an enemy of himself.)

Also:

1. If $x * y$ and $y * z$, then $x \sim z$ (A common enemy makes two people friends.)
2. If $x \sim y$ and $y * z$, then $x * z$ (An enemy of a friend is an enemy.)

Operations `setFriends(x,y)` and `setEnemies(x,y)` must preserve these properties.

Input Specification

The first line contains a single integer, n, the number of people.

Each of the following lines contains a triple of integers, c, x, y, where c is the code of the operation:

$c = 1$, `setFriends`

$c = 2$, `setEnemies`

$c = 3$, `areFriends`

$c = 4$, `areEnemies`

and x and y are its parameters, which are integers in the range $[0, n)$, identifying two (different) people.

The last line contains `0 0 0`.

All integers in the input file are separated by at least one space or line break. The only constraint is $n < 10\,000$, the number of operations is unconstrained.

Output Specification

For every `areFriends` and `areEnemies` operation, write `0` (meaning no) or `1` (meaning yes) to the output. Also, for every `setFriends` or `setEnemies` operation which contradicts with previous knowledge, output a `-1` to the output; note that such an operation should produce no other effect and execution should continue. A successful `setFriends` or `setEnemies` gives no output.

All integers in the output file must be separated by one line break.

Sample Input

```
10
1 0 1
1 1 2
2 0 5
3 0 2
3 8 9
4 1 5
4 1 2
4 8 9
1 8 9
1 5 2
3 5 2
0 0 0
```

Sample Output

```
1
0
1
0
0
-1
0
```

(Note that this problem is sourced from UVa.)