Problem A

The Monkey Dance

Input file: a.in
Output file: a.out
Source file: n:\a\a.{c, cpp, pas}

The director of Hind Circus has decided to add a new performance, the monkey dance, to his show. The monkey dance is performed simultaneously by *N* monkeys. There are *N* circles drawn on the ground, labelled from 1 to *N*. In the beginning, each monkey sits on a different circle. There are *N* arrows drawn from circle to circle in such a way that there is exactly one arrow pointing toward each ring and exactly one arrow pointing away from each ring.

When the show begins, at each whistle of the ringmaster, all the monkeys simultaneously jump from their respective circles to other circles following the arrows from their respective current circles. This is one step of the dance. The dance ends when *all* the monkeys have returned to the circles where they initially started. How long does the dance last?

Input

The input may have multiple cases. Each case consists of the value of N (1 <= N <= 100) on a line, followed by N lines, each with a pair of integers between 1 and N. The pair I_1, I_2 means that there is an arrow from circle I_1 to circle I_2. The input ends with 0 for the value of N.

Output

For each case, simply output the number of steps in the dance. Each output should be on a separate line.

Sample Input

```
5
1 2
2 3
3 1
4 5
5 4
4
1 4
2 3
4 1
3 2
0
```

Sample Output

```
6
2
```

Problem B

Coke or Chocolate Milk

Input file: b.in
Output file: b.out
Source file: n:\b\b.{c, cpp, pas}

Anybody who has been to a child's birthday has seen the following scenario:

- Parent: *OK Kids, what do you want to drink, Coke or Chocolate Milk?*
- Jamie: *Coke*
- John: *I hate Coke*
- Mary: *I want what John is having*
- Barry: *Ick! Then* I *don't want it if she's going to have it*
- *... etc.*

This is a large party. 1000 people have been invited and nearly that number may show up. Can you make everybody happy?

The input may contain several test cases. The first line of each test case is an integer N, the number of requests that must be satisfied. Following this line are the N requests, each on its own line. Each request has one of these formats:

```
   <person> wants <drink>
   <person> hates <drink>
   <person> wants same as <person>
   <person> wants different from <person>
   <person> wants <drink> if <person> gets <drink>
```

<person> is the name of a person - up to 20 lower-case letters. *<drink>* is either *Coke* or *chocolate milk.*

If it is possible to make everybody happy, print one line per person, in alphabetical order, with the format

```
   <person> gets <drink>
```

Be sure to correctly capitalize the drinks (Coke, chocolate milk).

If there is more than one way to make everybody happy (fat chance!), fill their glasses in alphabetical order and pour Coke (it's cheaper) whenever there is a choice. If it is not possible to make everybody happy, print

```
   Everybody gets water
```

Output a blank line after each test case. The input ends with 0 for the value of N.

Sample Input

```
4
jamie wants Coke
john hates Coke
mary wants same as john
```

```
barry wants different from mary
0
```

Output for Sample Input

```
barry gets Coke
jamie gets Coke
john gets chocolate milk
mary gets chocolate milk
```

Problem C

Partitions

Input file: c.in
Output file: c.out
Source file: n:\c\c.{c, cpp, pas}

Given a positive integer k, a *partition* is a sequence of positive integers in decreasing order whose sum is k.
For example, (12), (2,2,2,2,2,2) and (5,3,2,1,1) are all partitions of 12.
Given two distinct partitions, $(a_1,a_2,...,a_n)$ and $(b_1,b_2,...,b_m)$, we will say that $(a_1,a_2,...,a_n) > (b_1,b_2,...,b_m)$ if, for the smallest positive integer t such that $t <= n$, $t <= m$ and , we have $a_t > b_t$.

This ordering lets us put all the partitions of a given integer k in *lexicographical* order, where each partition in the ordering is greater than all the partitions before it.

For example, if $k = 5$, the partitions in lexicographical order are

(1,1,1,1,1)
(2,1,1,1)
(2,2,1)
(3,1,1)
(3,2)
(4,1)
(5)

Given k and a positive integer a, you are to find the a^{th} partition in the list of partitions of k sorted in lexicographical order.

The input will consist of a line with N, the number of test cases, followed by N lines, each of the form k a, where k and a are positive integers.

For each test case, your program should output the a^{th} partition in the list of partitions of k, or, if a is greater than the number of partitions of k, output "Too big".

Sample Input

```
3
1 1
5 4
5 8
```

Output for Sample Input

```
(1)
(3,1,1)
Too big
```