Overview

Problem	paintree	break	bad
Source	paintree.c paintree.cpp	N/A	bad.c bad.cpp
Input file	stdin	break.in	stdin
Output file	stdout	break.out	stdout
Time limit	1 second	N/A	1 second
Memory limit	64MiB	N/A	64MiB
Number of tests	10	10	10
Points per test	10	10	10
Detailed feedback	No	No	No
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Painting Trees

Keegan Carruthers-Smith and Max Rabkin

Introduction

Having painted all the white roses red, the Five, Seven and Two of Spades have decided to paint the rose tree, too.

Task

The tree consists of N branches numbered 1 to N. Branch 1 is the trunk and does not have a parent; every other branch has a "parent", whose number is smaller than its own.

The gardeners will only paint one branch at a time (even though there are three of them) and to make sure they don't miss a branch, they will only paint a branch if its parent is already painted.

Five claims there are forty-two orders in which they can paint the branches. Seven says this is obviously rubbish, and that the answer is *fifty-two*. Two has asked you to calculate the correct number of orders. Since they only want to determine which of them (if any) are right, you only need to find the answer modulo 10 007.

Example

If the tree has a trunk which splits into two branches (2 and 3), and 2 has a child 4 then there are three orders the gardeners can paint the tree:

- $1, 2, 3, 4$
- $1, 2, 4, 3$
- $1, 3, 2, 4.$

Input (stdin)

The first line of input contains an integer N. For $i = 2, 3, \ldots, N$, line i contains a single integer, the number of the parent of branch i.

Sample input

```
4
1
1
2
```

Output (stdout)

The output consists of a single line containing single integer between 0 and 10 006, the number of ways the gardeners can paint the tree, modulo 10 007.

Sample output

```
3
```

Constraints

- $N \leq 1000$

Additionally, in 20% of the test cases: $N \leq 10$

Time limit

1 second.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

Prison Break

Carl Hultquist

Introduction

Many years ago, Bruce was falsely convicted for creating a computer worm that would syphon money from other people's bank accounts into his own. Since then, he's been locked up in prison, waiting for the right moment when he can make his escape and clear his name.

That moment has arrived: the guards are all distracted, and Bruce has managed to pick the lock to his cell! Now he just needs to get out of the prison...

To do this, Bruce will need to visit as many rooms in the cell-block as he can looking for the master key which will let him out the main door. The more rooms he can visit, the better his chances of finding the key. Since Bruce has been in the prison for a long time, he knows exactly how many rooms there are and how they're connected to each-other by doorways.

The rooms are also wired with motion sensors, which will go off as soon as Bruce walks through them: when they do, the guards will be alerted and will start running towards the room that set off the alarm. For this reason, Bruce **cannot** visit any room more than once, for if he did so he would likely run into the guards and be re-imprisoned!

Task

Your task is to find a route that Bruce should take so that he visits as many rooms in the cell-block as possible.

Example

Suppose the cell-block has the layout shown in figure 1.

The `B` indicates where Bruce is currently, and the numbers indicate doorways between rooms.

In this example, Bruce can visit all of the rooms by going through the following doorways (in order): 1, 3, 4, 5, 6, 7.

Input (break.in)

The first line of input will contain three integers, N, M and B. N is the number of rooms in the cell-block (the rooms are numbered from 1 to N), M is the number of doorways in the cell-block, and B is the number of the room in which Bruce starts. The next M lines will describe the doorways: each of these lines will contain 2 integers, I and J, indicating that there is a doorway between the two rooms with numbers I and J.

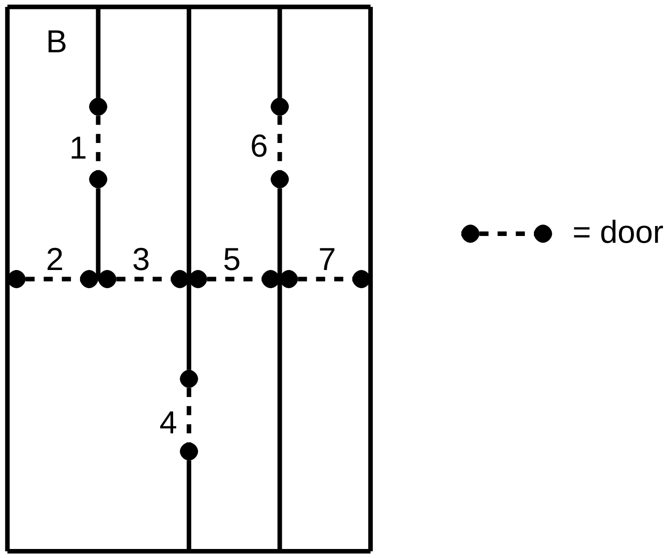

Figure 1: Example prison layout

Sample input

```
7 7 1
1 2
1 3
2 3
3 4
4 5
5 6
6 7
```

Output (break.out)

Your first line of output should contain a single integer, K, which is the maximum number of rooms that Bruce could visit (including the room that he starts in). On the next line you should then output $K-1$ integers describing the doorways (in order) that Bruce must pass through to visit this number of rooms (the doorways are numbered from 1 to M, and are given in this order in the input).

Sample output

```
7
1 3 4 5 6 7
```

Constraints

- $3 \leq N \leq 30$

- No room will have more than 4 doorways.

Scoring

If your output differs in any way from the format described above, then your program will score 0.

If the path you specify is invalid, you will score 0 points. If you have found a valid path for your value of K, then you will score $\frac{10}{1+(O-K)}$ where O is the actual maximum number of rooms that Bruce can visit. In the case that your score is not an integer, it will be rounded *downwards* to the nearest integer.

Bad Gates

Keegan Carruthers-Smith (from ACM ICPC World Finals 2009)

Introduction

Fred the manic storekeeper has recently expanded into the business of electronics. He just received a large shipment of logic circuits for a *very* good price.

A logic circuit maps its input through various gates to its output with no feedback loops in the circuit. The input and output are an ordered set of logical values, represented by ones and zeros. The circuits in the shipment are comprised of *and* gates (which output 1 only when their two inputs are both 1), *or* gates (which output 1 when one or both of their inputs are 1), *exclusive or* (xor) gates (which output 1 only when exactly one of the two inputs is 1), and *not* gates (which output the complement of their single input).

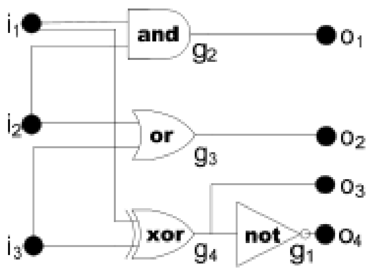

Figure 2: The circuit on the right represents the sample input.

The reason Fred got such a good deal on the shipment of circuits is that many circuits contain faulty gates. Luckily for Fred the gates *only* fail in one of three ways:

1. always inverting the correct output,
2. always yielding 0, or
3. always yielding 1

In fact Fred also has the guarantee that *at most one* gate is faulty in a logic circuit.

Task

Your task is to write a program that analyzes a circuit and a number of observations of its input and output to see if the circuit is performing correctly or incorrectly. If at least one set of inputs produces the wrong output, your program must also attempt to determine the unique failing gate and the way in which this gate is failing. This may not always be possible.

Example

The circuit in Figure 2 represents the sample input. The first observation is that with the inputs $i_1 = 0, i_2 = 1, i_3 = 0$ you get the outputs $o_1 = 0, o_2 = 1, o_3 = 0, o_4 = 1$. This observations does not indicate any faults since

$$\begin{aligned} o_1 &= g_2 \\ &= i_1 \ \&\& \ i_2 \\ &= 0 \ \&\& \ 1 = 0 \end{aligned}$$

$$\begin{aligned} o_2 &= g_3 \\ &= i_2 \ || \ i_3 \\ &= 1 \ || \ 0 = 1 \end{aligned}$$

$$\begin{aligned} o_3 &= g_4 \\ &= (i_1 \ != \ i_3) \\ &= (0 \ != \ 0) = 0 \end{aligned}$$

$$\begin{aligned} o_4 &= g_1 \\ &= (!g_4) \\ &= (!0) = 1 \end{aligned}$$

Notice that this observation did not pick up the fault, since g_2 is stuck on 0. But in the third observation we have i_1 and i_2 set to 1. If g_2 was not faulty it would output 1 and 1 = 1, but since it is stuck on 0 it outputs 0. This is the only fault which matches the observations so the output is “Gate 2 is failing; output stuck at 0”.

Input (stdin)

The input consists of a circuit with input and output descriptions. Each test case has the following parts in order.

1. A line containing three positive integers giving the number of inputs (N), the number of gates (G), and the number of outputs (U) in the circuit.
2. One line of input for each gate. The first line describes gate g_1. If there are several gates, the next line describes gate g_2, and so on. Each of these lines contains the gate type (a = *and*, n = *not*, o = *or*, and x = *exclusive or*), and identification of the input(s) to the gate. Gate input comes from the circuit inputs ($i_1, i_2, \dots$) or the output of another gate ($g_1, g_2, \dots$).
3. A line containing the numbers of the gates connected to the U outputs $u_1, u_2, \dots$. For example, if there are three outputs, and u_1 comes from g_5, u_2 from g_1, and u_3 from g_4, then the line would contain: 5 1 4

4. A line containing an integer which is the number of observations of the circuits behavior (B).
5. Finally B lines, each containing N values (ones and zeros) giving the observed input values and U values giving the corresponding observed output values. No two observations have the same input values.

Consecutive entries on any line of the input are separated by a single space.

Sample input

```
3 4 4
n g4
a i1 i2
o i2 i3
x i3 i1
2 3 4 1
4
0 1 0 0 1 0 1
0 1 1 0 1 1 0
1 1 1 0 1 0 1
0 0 0 0 0 0 1
```

Output (stdout)

There are 5 different possibilities of output:

1. `No faults detected`
2. `Unable to totally classify the failure`
3. `Gate` i `is failing; output stuck at 1`
4. `Gate` i `is failing; output inverted`
5. `Gate` i `is failing; output stuck at 0`

If no faults are detected from the observations, then output the first string. If it is impossible to detect which gate is failing or how a gate is failing, output the second. Otherwise if you can infer that gate g_i is failing and how, output the corresponding output from 3 to 5.

Sample output

```
Gate 2 is failing; output stuck at 0
```

Constraints

- $1 \leq N \leq 8$
- $0 \leq G \leq 19$
- $0 \leq U \leq 19$
- $0 \leq B \leq 19$

Time limit

1 second.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.