Problem K
Magically Marked Matching Master
Time limit: 2 seconds

This is a multi-pass, interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

The concept of the game, illustrated by ChatGPT.

In the classical *online matching* game, edges of a graph are revealed one by one. Each time an edge is shown, you must immediately and irrevocably decide whether to include it in your matching. Your goal is to ensure that no two chosen edges share a vertex while maximizing the total number of edges selected.

You are an ambitious computer scientist, hungry for fame. As you well know, finding a maximum matching in an online setting is provably impossible with high probability — even for the simplest classes of graphs like bipartite graphs. Yet where others see a fundamental barrier, you see an opportunity for glory. Today, you claim to have cracked the long-standing open problem for trees and are ready to unveil your groundbreaking algorithm to the world.

There is just one problem: yesterday, you discovered a bug in your proof, and you still have no idea how to fix it. Unwilling to abandon the spotlight, you decide to turn your demonstration into a magic show — your assistant, hidden backstage, will transmit *a little* hint about the entire tree with n vertices before the game begins.

Your original plan was simple: have your assistant send the answer to each edge query directly. To this end, you have installed a secret earpiece capable of receiving a single binary string of length $(n-1)$. But the audience is skeptical of your "breakthrough." To prevent any pre-arranged trickery, they demand that the edges be presented in a uniformly random order — one that neither you nor your assistant can predict or control.

The stage is set. The lights are on you. Now prove your claim — without changing the channel.

Interaction Protocol

Your solution is executed twice on each test. In the `prepare` round, your solution will act as the assistant. In the `play` round, your solution will act as *you*, the magician. In either of the rounds, your solution will be evaluated as an interactive procedure.

Prepare Round

The first line of the input contains the word `prepare`. The second line contains an integer n ($2 \le n \le 500$) — the number of vertices. The following $(n-1)$ lines each contain two integers u and v ($1 \le u, v \le n$, $u < v$), denoting an undirected edge between u and v in the tree.

You must output one line containing a binary string s of exactly length $(n-1)$ consisting of characters `0` and `1` only. This is the string transmitted to the second `play` round. If your output does not align with the desired format, you will receive the `Wrong Answer` verdict.

Play Round

Your solution will be restarted for the `play` round.

The first line contains the word `play`. The second line contains the integer n. The third line contains the binary string s exactly as you printed it in the `prepare` round.

After that, the game starts with $(n-1)$ turns. In each turn, a line containing two integers u and v ($1 \le u, v \le n$, $u < v$) will be provided. As a response, you should output one line containing either `take` or `ignore`. All edges in the first round will be provided exactly once, and the order of the edges is chosen uniformly at random from $(n-1)!$ possible permutations.

When there is an output in invalid format, the game will end immediately with no further information from the interactor.

Evaluation of Correctness

Let M be the set of edges your solution takes. It is considered acceptable if no vertex is incident to more than one taken edge and $|M| = |M^*|$, where M^* is the maximum matching of the graph.

There are exactly 50 tests, excluding the sample tests. In order to pass the problem, your solution should be correct on all of the tests, including the sample tests.

All shuffles are guaranteed to be fixed before receiving the solution's output. The randomness is fixed for each test, implemented by fixing the seed of a pseudorandom number generator.

A testing tool is provided to help you develop and test your solution.

Read **Sample 1, Pass 1** **Write**

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

```
00
```

Read **Sample 1, Pass 2** **Write**

```
play
3
00
1 3
```

```
take
```

```
1 2
```

```
ignore
```

Explanation of Sample 1: Since $|M^*| = 1$, taking any edge is acceptable, and the hint can be arbitrary.