Artificium

Boat Race Analysis

Runtime Limit – 3s

Problem Statement

Hello engineer,

We need you to help us find the order of boat rankings. Analyzes of the current race are at your disposal.

Your objective will be to process this information to make it visual to the spectators who follow the race live.

Race analysis data can take two forms:

- *Boat_a* is in front of *Boat_b*
- *Boat_a* is not ahead of *Boat_b*

To make this information visual you will have to determine the position of the boats, then display in **alphabetical order (case-insensitive)** the position of the boats in the form "_ *Boat_name*" with more underscores to indicate that a boat is further ahead.

Be careful, sometimes the analyzes are incorrect and can lead to impossible positioning of the boats. You will need to detect circular references to deal with these cases.

When you detect a circular reference error due to a bad reading of the name of the boat :

- No boat positions have to be output
- You will have to display all the analysis affected by the error in the order of appearance.

If after analysis a boat can have several possible positions, we will retain the smallest. The smallest position is the one that will show the fewest "_ " markers before the boat name.

Format

Input

Line 1: The number *n* of analysis extracted by your team from the camera images.

Next *n* lines: an *analysis* in the form : "*a* is in front of *b*" **or** "*a* is not ahead of *b*"

Output

Either: position of the boats, one line per boat.

Or: analysis affected by circular reference error, one line per analysis.

Constraints

length of *Boat_name* ≥ 1

Boat_name always starts with a letter.

Boat_name may contain alphanumeric and special characters as well as space characters.

Each circular reference error affects at most two lines (but there can be more than one circular reference error in a case).

Sample 1

Input

```
2
E is in front of B
B is in front of F
```

Output

```
_ B
_ _ E
F
```

Sample 2

Input

```
2
e is not ahead of f
e is in front of f
```

Output

```
e is not ahead of f
e is in front of f
```