Task 13: **Clu{e}**

[450 points]

A popular murder mystery board game known as Clue requires that players determine the murderer, the means of the murder, and the location of the murder based on given "clues." Players do this by going through turns, where they gain bits of knowledge and deduce others, until they finally arrive at the solution.

Problem Statement

Write a functional engine to determine the details of a murder at the earliest convenience, given a set of "clues."

Input Format

Your Engine will be fed 3 types of input in the following order:

1. `<Data>` : All games of Clue have a fixed number of suspects, weapons, and locations. These will be fed as the first 3 lines of the input. They will be given as three lists and will take the form of `{[ListName] : [data] [data] [data] …}`
2. `<Rule>` : Rules are logical assumptions that the engine can make to build its knowledge database. Rules will always be fed as the second part of the input. Rules will be given in the format `{[object], [object] => [object]}`. Each object will be taken from a different one of the three data lists. If the first two objects are found to be true, the third can be assumed true. The inverse works as well: if the first two objects are found to be false, the third can be assumed false.
3. `<Clue>` : The engine should take in "clues," which will be indicated by the form `{[yes/no] [object]}`. If the clue is an affirmative, i.e. `{[yes] [object]}`, then you can assume that object was the one used. If the clue is a negative, i.e. `{[no] [object]}`, then you can assume that object was not used and can be ignored.

Output Format

The output should list the solution in the following form: `{[ClueNum] [Suspect] [Weapon] [Location]}`, where `ClueNum` is the clue that the puzzle was solved with (because we wish to solve it at earliest possible convenience). `ClueNum` starts at 1.

Hint: First figure out how to handle input, because you don't know the number of entries in the first three lists or the number of rules that follow.

Sample Input

```
{[Suspects] : [Darth Vader] [Luke] [Han Solo] [Yoda]}
{[Weapons] : [Force] [LightSaber] [Blaster]}
{[Locations] : [Mos Eisley Cantina] [Death Star] [Dagobah] [Death Star II] [Echo Base]}
{[Han Solo], [Mos Eisley Cantina] => [Blaster]}
{[Force], [Death Star II] => [Darth Vader]}
{[Han Solo], [LightSaber] => [Echo Base]}
{[Yoda], [Dagobah] => [Force]}
{[LightSaber], [Death Star] => [Darth Vader]}
{[yes] [Death Star II]}
{[no] [Blaster]}
{[no] [Luke]}
{[yes] [Darth Vader]}
{[no] [Force]}
{[no] [Dagobah]}
{[no] [Yoda]}
{[yes] [LightSaber]}
```

Sample Output

```
{[4] [Darth Vader] [LightSaber] [Death Star II]}
```

Sample Test Case Explanation

The first clue rules out all locations other than `Death Star II`, while the fourth rules out all suspects other than `Darth Vader`. With this information, our engine now knows that `Yoda` is not the suspect and that `Dagobah` is not the location. Since it knows these 2 pieces of information, it can use rule 4 to determine that the `Force` is not the weapon. We have already ruled out the `Blaster` as the weapon thanks to clue 2, so we know the weapon is the `LightSaber`. Thus, we have solved the mystery using clue 4, so we output the solution.