Peanuts and pretzels

My favourite bar puts out bowls of peanuts and pretzels for patrons to snack on (and to encourage further beverage purchases). Over time, a strange set of customs has developed when two people share a table there.

- Snacks must be taken strictly in turn (*i.e.*, after you've taken a snack you must wait for your companion to take one before you can do so again).
- It is always permissible to take either exactly one peanut, or exactly one pretzel.
- Other combinations may be allowed—these change from day to day and are listed on a board behind the bar.
- If it is your turn to take a snack and both bowls are empty, you must go to the bar to have them refilled and buy the next round of drinks.

The format for the special combinations allowed is that each consists of a pair of expressions, the first indicating the number of peanuts and the second the number of pretzels. An expression can be any one of:

- $= k$, where k is a non-negative integer, *e.g.*, `=3`;
- $< k$, where k is a non-negative integer, *e.g.*, `<3`;
- $> k$, where k is a non-negative integer, *e.g.*, `>3`.

The interpretation of such rules is described in the following examples:

- `=3 =2` you may take exactly three peanuts and exactly two pretzels;
- `<3 =2` you may take fewer than three peanuts, *i.e.*, 0, 1 or 2, and exactly two pretzels;
- `=3 >2` you may take exactly three peanuts and more than two pretzels;
- `<3 <2` you may take fewer than three peanuts and fewer than two pretzels.

All snacks must consist of at least one item, so in the last case saying "I'm on a diet" does not count as taking a snack.

Since drinks are very expensive, I always try to ensure that I take the last snack even if it means consuming rather more than I'd really like to. However, lately I've noticed that I seem to be buying most of the rounds—and

that my drinking buddies are consulting some sort of app before making their snack choices. What's going on?

Task

Your task is to develop the heart of that app. Given today's rules and the number of peanuts and pretzels in each bowl, determine what choice should be made (if possible) to ensure that eventually you get the last snack.

Input will come from the standard input stream. It will consist of a sequence of scenarios, separated from one another by single blank lines. A scenario consists of a number of lines. The first line describes the number of peanuts and the number of pretzels (in that order). There could be up to 1000 peanuts. There could be up to 1000 pretzels. The remaining lines describe the special rules applying today for what constitutes a single snack in the format shown above.

Output for each scenario should be a legal snack which, no matter how your drinking companion responds, will lead to you eventually taking the last item. Remember it is always permissible to take exactly one peanut or exactly one pretzel. If there is no snack which gives you a guaranteed free drink, then the output should be 0 0.

Example

Input:

```
3 3
>1 <3

5 5
=2 =2
=3 =3
=2 =3
=3 =2
```

Output:

```
2 2
0 0
```

In the first example, the `2 2` move leaves `1 1` where the only allowed move is to take a single item, and then you can get the last one.

In the second example, if you use any of the special moves, your companion can use another one and get a free drink. But if you use an ordinary move to `4 5` or `5 4`, then `3 2` or `2 3` leaves `1 3` or `3 1` from which no special moves are allowed, and again you will eventually find yourself facing two empty bowls.

(Group 2)

Notes

The *problem* is a theory of combinatorial games problem, related to but distinct from Nim. There's a great deal of theory about combinatorial games. Berlekamp, Conway, and Guy's classic book, "Winning Ways for your Mathematical Plays", is in our library. PDFs of older editions can be found on-line. None of that is needed for this étude. (People have figured out how to derive error-correcting codes from some games. *Everything* connects.)

The *technique* you need to solve the problem is a technique you will find useful over and over again. In fact, there are several approaches you could take. Since each move *must* reduce the number of peanuts or the number of pretzels, you know that a recursive procedure is going to work.

Testing

As usual, you want to test your program in cases where you know what the right answer is. You might, for example, consider cases with no peanuts or no pretzels. You might also consider cases that reduce to Nim, because you'll be able to find strategies for Nim easily enough.