Extreme Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Many problems given in programming competitions are extreme in this or that regard. Examples include:

- a problem that is solved by doing lots of heavy math on paper and then by printing one well-known number, rounded to the number of digits given in the input file;
- a problem that spans more than four pages and requires you to write a simulation system that tracks multiple skills of several secret agents and chooses the best combination of them for each mission;
- a problem for which it is a proven fact that no correct solution will ever exist, but it was still given in a contest.

This time you are given a problem that is also extreme, in that it has only eight possible tests. And, of course, it will be about something extreme.

We consider functions of two integer variables, defined on the $[-10; 10] \times [-10; 10]$ square. It means that you can perform a call to $f(x, y)$ only if x and y are integers and $-10 \le x, y \le 10$. Such a function f is said to have a *local minimum* at (x, y) if the following statements hold simultaneously:

- $f(x, y) < f(x - 1, y)$;
- $f(x, y) < f(x + 1, y)$;
- $f(x, y) < f(x, y - 1)$;
- $f(x, y) < f(x, y + 1)$.

A *local maximum* is defined in a similar way, only the inequalities are reversed. A function f is said to have a *plateau* at (x, y) if at least one of the following statements holds:

- $f(x, y) = f(x - 1, y)$;
- $f(x, y) = f(x + 1, y)$;
- $f(x, y) = f(x, y - 1)$;
- $f(x, y) = f(x, y + 1)$.

Note that all the function invocations above must happen on the points from the function domain, that is, the $[-10; 10] \times [-10; 10]$ square. In particular, this means that a point on the boundary of this square **cannot be** a local maximum or a local minimum, but can still be a plateau.

You need to find a function which has — or does not have, depending on the information in the input:

- multiple local maxima;
- multiple local minima;
- some plateaus.

Note that "multiple" means "at least two". Also note that your function shall be defined in a specific way; see the Output section for details.

Input

The input consists of three lines.

- The first line starts with "`Multiple local maxima: `" and ends with either "`Yes`" or "`No`". This specifies whether your function shall or shall not have multiple local maxima.
- The second line starts with "`Multiple local minima: `" and ends with either "`Yes`" or "`No`", and similarly deals with local minima.
- The third line starts with "`Plateaus: `" and also ends with either "`Yes`" or "`No`". This specifies whether your function shall or shall not have plateaus.

Output

The output shall consist of one line that defines your function.

If no such function can be represented as per the format below, print "`No solution`".

Otherwise, print your function using the reverse Polish notation. Recall that the reverse Polish notation is a way to describe a function using some sort of a stack machine: it is a sequence of operations, some of which push values onto the stack, while some pull a few values from the top of the stack, perform some math and push the result back to the stack.

Your function shall contain at most 1000 tokens, separated by single spaces, where each token is one of the following.

- An integer constant ranging from `-9` to `+9`. This will push the corresponding number onto the stack.
- A variable, either `x` or `y`. This will push the value of that variable onto the stack.
- An operation, which can be `+`, `-`, `*`, or `^`. The asterisk means multiplication, whereas the `^` character means raising to the power. Each of these operations will take two numbers from the stack, apply the operation and push the result back to the stack. The evaluation order is such that "`9 5 -`" evaluates to 4, and similarly "`x 2 ^`" evaluates to x^2. As a special case, 0^0 evaluates to 1.

Note that whenever one of the following things happens:

- an operation attempts to take a number from an empty stack;
- the `^` operation attempts to raise something to a negative power;
- the result of an operation overflows the 32-bit signed integer;
- or at the end of the evaluation the size of the stack is not equal to one —

you receive a Wrong Answer outcome for the test where it happened.

Examples

standard input	standard output
Multiple local maxima: No Multiple local minima: No Plateaus: No	x 3 - 4 ^ y -5 + 2 ^ +
Multiple local maxima: No Multiple local minima: No Plateaus: Yes	1

Note

The example answer to the first test encodes the function $(x-3)^4+(y+(-5))^2$. Note that it has no local maxima, no plateaus, and just one local minimum.