Problem B
Equals are Equals
Input: expressions.txt

Mr. Simpson got up with a slight feeling of tiredness. It was the start of another day of hard work. A bunch of papers were waiting for his inspection on his desk in his office. The papers contained his students' answers to questions in his Math class, but the answers looked as if they were just stains of ink.

His headache came from the "creativity" of his students. They provided him a variety of ways to answer each problem. He has his own answer to each problem, which is correct, of course, and the best from his aesthetic point of view.

Some of his students wrote algebraic expressions equivalent to the expected answer, but many of them look quite different from Mr. Simpson's answer in terms of their literal forms. Some wrote algebraic expressions not equivalent to his answer, but they look quite similar to it. Only a few of the students' answers were exactly the same as his.

It is his duty to check if each expression is mathematically equivalent to the answer he has prepared. This is to prevent expressions that are equivalent to his from being marked as "incorrect", even if they are not acceptable to his aesthetic moral.

He had now spent five days checking the expressions. Suddenly, he stood up and yelled, "I've had enough! I must call for help."

Your job is to write a program to help Mr. Simpson to judge if each answer is equivalent to the "correct" one. Algebraic expressions written on the papers are multi-variable polynomials over variable symbols $a, b, ..., z$ with integer coefficients, e.g., $(a + b^2)(a - b^2)$, $ax^2 + 2bx + c$ and $(x^2 + 5x + 4)(x^2 + 5x + 6) + 1$.

Mr. Simpson will input every answer expression as it is written on the papers; he promises you that an algebraic expression he inputs is a sequence of terms separated by additive operators '`+`' and '`-`', representing the sum of the terms with those operators, if any; a term is a juxtaposition of multiplicands, representing their product; and a multiplicand is either (a) a non-negative integer as a digit sequence in decimal, (b) a variable symbol (one of the lowercase letters '`a`' to '`z`'), possibly followed by a symbol '`^`' and a non-zero digit, which represents the power of that variable, or (c) a parenthesized algebraic expression, recursively. Note that the operator '`+`' or '`-`' appears only as a binary operator and not as a unary operator to specify the sign of its operand.

He says that he will put one or more space characters before an integer if it immediately follows another integer or a digit following the symbol '`^`'. He also says he may put spaces here and there in an expression as an attempt to make it readable, but he will never put a space between two consecutive digits of an integer. He remarks that the expressions are not so complicated, and that any expression, having its '`-`'s replaced with '`+`'s, if any, would have no variable raised to its 10th power, nor coefficient more than a billion, even if it is fully expanded into a form of a sum of products of coefficients and powered variables.

Input

The input to your program is a sequence of blocks of lines. A block consists of lines, each containing an expression, and a terminating line. After the last block, there is another terminating line. A terminating line is a line solely consisting of a period symbol.

The first expression of a block is one prepared by Mr. Simpson; all that follow in a block are answers by the students. An expression consists of lowercase letters, digits, operators '+', '-' and '^', parentheses '(' and ')', and spaces. A line containing an expression has no more than 80 characters.

Output

Your program should produce a line solely consisting of "yes" or "no" for each answer by the students corresponding to whether or not it is mathematically equivalent to the expected answer. Your program should produce a line solely containing a period symbol after each block.

Sample Input

```
a+b+c
(a+b)+c
a-(b-c)+2
.
4ab
(a - b)(0-b+a) - 1a ^ 2 - b ^ 2
2 b 2 a
.
108 a
2 2 3 3 3 a
4 a^1 27
.
.
```

Output for the Sample Input

```
yes
no
.
no
yes
.
yes
yes
.
```