Problem E. Infallibly Crack Perplexing Cryptarithm

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are asked to crack an encrypted equation over binary numbers.

The original equation consists only of binary digits (‘0’ and ‘1’), operator symbols (‘+’, ‘-’, and ‘*’), parentheses (‘(’ and ‘)’), and an equal sign (‘=’). The encryption replaces some occurrences of characters in an original arithmetic equation by letters. Occurrences of one character are replaced, if ever, by the same letter. Occurrences of two different characters are never replaced by the same letter. Note that the encryption does not always replace all the occurrences of the same character; some of its occurrences may be replaced while others may be left as they are. Some characters may be left unreplaced. Any character in the English alphabet, in either lowercase or uppercase, may be used as a replacement letter. Note that cases are significant, that is, ‘a’ and ‘A’ are different letters. Note that not only digits but also operator symbols, parentheses and even the equality sign are possibly replaced.

The arithmetic equation is derived from the start symbol of Q of the following context-free grammar.

$$
\begin{array}{rcl}
Q & ::= & E = E \\
E & ::= & T \mid E + T \mid E - T \\
T & ::= & F \mid T * F \\
F & ::= & N \mid \texttt{-}F \mid \texttt{(}E\texttt{)} \\
N & ::= & \texttt{0} \mid \texttt{1}B \\
B & ::= & \varepsilon \mid \texttt{0}B \mid \texttt{1}B
\end{array}
$$

Here, ε means empty.

As shown in this grammar, the arithmetic equation should have one equal sign. Each side of the equation is an expression consisting of numbers, additions, subtractions, multiplications, and negations. Multiplication has precedence over both addition and subtraction, while negation precedes multiplication. Multiplication, addition and subtraction are left associative. For example, x`-`y`+`z means `(`x`-`y`)+`z, not x`-(`y`+`z`)`. Numbers are in binary notation, represented by a sequence of binary digits, `0` and `1`. Multi-digit numbers always start with `1`. Parentheses and negations may appear redundantly in the equation, as in `((--`x`+`y`))+`z.

Write a program that, for a given encrypted equation, counts the number of different possible original correct equations. Here, an equation should conform to the grammar, and it is correct when the computed values of the both sides are equal.

For Sample Input 1, `C` must be ‘`=`’ because any equation has one ‘`=`’ between two expressions. Then, because `A` and `M` should be different, although there are equations conforming to the grammar, none of them can be correct.

For Sample Input 2, the only possible correct equation is “`-0=0`”.

For Sample Input 3 (“B-A-Y-L-zero-R”), there are three different correct equations, “`0=-(0)`”, “`0=(-0)`”, and “`-0=(0)`”. Note that one of the two occurrences of zero is not replaced with a letter in the encrypted equation.

Input

The input consists of a single test case which is a string of English alphabet characters, binary digits, operator symbols, parentheses and equal signs. The input string has at most 31 characters.

Output

Print in a line the number of correct equations that can be encrypted into the input string.

Examples

standard input	standard output
ACM	0
icpc	1
BAYLOR	3
-AB+AC-A	1
abcdefghi	0
111-10=1+10*10	1
0=10-1	0