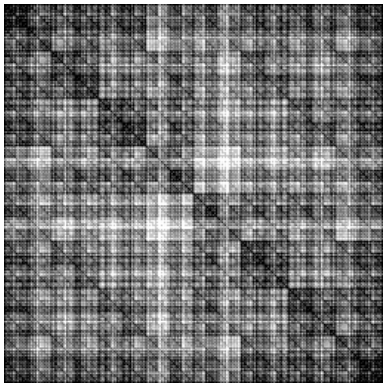
Problem G

Paper



Nils is preparing his Master's thesis, and wants to include the source code for his new logarithmic-time halting problem solver. To minimize the environmental impact of the inevitable millions of printouts of his revolutionary paper around the world, he wants to make his program as short as possible.

You will help by designing an expression optimizer for boolean expressions. The input will be a fully parenthesized expression, using the standard one-character boolean operators (`&`, `|` and `!`).

For each expression, print the length of the shortest equivalent expression (not including spaces).

```
expression ::= variable
             | '(' expression ' ' operator ' ' expression ')'
             | '!' expression
operator ::= '&' | '|'
variable ::= 'x' | 'y' | 'z'
```

Note in particular that parentheses are required for all operators. This means that each operator will contribute three characters to the length, whereas NOTs and variable references contribute one character each.

Input specifications

The input has $1 \leq n \leq 50$ cases, where n is given by the first line of input. Each test case is given by a line with an expression as decribed earlier. Each expression will be no more than 500 characters long (including spaces).

Output specifications

For each test case output the length of the shortest equivalent expression.

In the example, the first expression is never true. It can be written as `(x & !x)`, which has a length of 6. The third example can be simplified by applying De Morgan's law twice (after which it becomes equivalent to the second example).

Sample input

```
3
(((x & !y) & (y & !z)) & (z & !x))
((x & z) | (y & z))
!((!x & !y) | !z)
```

Output for sample input

```
6
9
9
```