Problem 5—Logical Evaluation

Although Bruce Wayne was actually an excellent student in college (studying criminology and psychology), he does not evidently make much use of his expensive education these days, in that he was able to build his philanthropic organization, the Wayne Foundation, entirely with inherited wealth. However, despite the image his bored, womanizing persona might convey, Bruce's deductive mind is still just as sharp as it was in college. Bruce loves to play the logic games in the newspaper, and, while he could write a logical evaluation parser quite easily, he has left the task to you.

A logical statement may be:
A single capital letter (with assigned value of either TRUE (1) or FALSE (0))
~α (with value the negation of α)
(α|β) (with value the logical (inclusive) or of α and β)
(α&β) (with value the logical and of α and β)

α and β represent full valid logical expressions in the above specification.
Notice that | and & are ALWAYS surrounded by parentheses in these expressions. As a result, order of operations plays no part in the evaluation of logical expressions; each logical expression simplifies to a unique boolean value.

You are to write a program to evaluate a logical expression. Bruce would do it himself but his other interests are presently driving him...

(wait for it... wait for it...)

...batty.

INPUT SPECIFICATION. Each data case consists of two lines. The first line contains exactly 26 characters followed by <**EOLN**>. Each of these characters will be a zero or one and corresponds to the logical values of the capital letters. The first character is the logical value of A; the second is the logical value of B; and so forth.
The second line consists of a valid logical expression as defined above, followed by <**EOLN**>. The expression will not contain any syntax errors. There will be no embedded spaces. There will be no extra parentheses (beyond those as defined in the expression). There will be no dirty tricks. The last data case will be followed by <**EOF**>.

OUTPUT SPECIFICATION. The output cases should appear in the same order as their respective input cases. The output should be in the format "Case c: TRUE" or "Case c: FALSE" depending on the evaluation of the logical expression. C is the case number. Each output case should be followed by exactly one <**EOLN**>.

SAMPLE INPUT.

```
11111111111111111111111111<EOLN>
((P&~Q)|(Q&~P))<EOLN>
00000000000000000000000000<EOLN>
~P<EOLN>
00000000000000000000000000<EOLN>
~(~P&~Q)<EOLN>
<EOF>
```

SAMPLE OUTPUT.

```
Case·1:·FALSE<EOLN>
Case·2:·TRUE<EOLN>
Case·3:·FALSE<EOLN>
<EOF>
```