Problem 2—APL Expressions

Westeros has always been a bit behind North America when it comes to developing programming languages. In fact, although this is seldom seen, the primary programming language used in Westeros is APL, a language that was pretty much dead here by 1980. The weird thing about APL is that its expressions looked like the expressions found in virtually every other language such as C or Java or BASIC, but there was no "order of operations". Every operation had equal priority to every other, but—and this is where it gets weird—the operations were processed from right to left! For example, 5*10–2 would evaluate to 40 in APL, because 10–2 = 8 and 5*8 = 40. You can use parentheses to override the normal evaluation order just as you can in most major languages. You are to write a program to evaluate APL expressions.

INPUT SPECIFICATION. Each input case contains a properly formatted and valid APL expression followed by **<EOLN>**. Spaces will not appear in the expression. The only operations present in the expression will be +, –, and *, and the only numbers that will appear are integers. You may assume that the numbers will be small enough so that normal, sane operations on these integers will not cause an overflow. Negative numbers will never be given in the expression, but the answer might very well be negative. An extra **<EOLN>** will follow the last case. It is not to be processed; it just signals the end of input.

OUTPUT SPECIFICATION. The output cases are to be processed in the same order as the input cases. Each case will be of the form "Case *c*: *e* evaluates to *n*." where *c* is the case number, *e* is the input expression, and *n* is the answer. Two **<EOLN>** characters should follow each case.

SAMPLE INPUT.

```
5*10-2<EOLN>
(5*10)-2<EOLN>
<EOLN>
<EOF>
```

SAMPLE OUTPUT.

```
Case·1:·5*10-2·evaluates·to·40.<EOLN>
<EOLN>
Case·2:·(5*10)-2·evaluates·to·48.<EOLN>
<EOLN>
<EOF>
```