ECOO '12 R3 P3 - Steam Arithmetic

Time limit: 1.0s   **Memory limit:** 64M

Steam is a brand new dialect of Lisp, one of the oldest high-level programming languages. The name Lisp comes from the phrase "List Processing" because the only structure in Lisp (and therefore in Steam) is the list. A list is a sequence of tokens separated by whitespace and enclosed in round brackets. Here is a list with 6 elements: `(6 z 4.2 hello 4 world)`.

Lists can be nested, meaning they can have other lists as elements. Here is an example of a nested list: `(5 (x t e d) (6 7 (-1 2)))`. This list has three elements. The first is `5`, the second is the list `(x t e d)`, and the third is the list `(6 7 (-1 2))`, which also has another list nested inside of it. There is no limit to how many lists can be nested inside other lists.

Arithmetic in Steam uses "prefix" notation, in which the operator is written before the operands. We are used to seeing arithmetic in "infix" notation like this: $5 + 3$. But in prefix notation `5 + 3` becomes `+ 5 3`. In Steam, arithmetic expressions are lists like this:

$$\begin{array}{ll}
\texttt{(+ 5 3)} & \text{means } 5 + 3 \\
\texttt{(* 7 2)} & \text{means } 7 \times 2 \\
\texttt{(* (- 5 2) (+ 2 3))} & \text{means } (5 - 2) \times (2 + 3)
\end{array}$$

In Steam arithmetic, there are 5 operators: `+`, `-`, `*`, `q` (for quotient) and `r` (for remainder, which is the same as the standard "modulus" or $\bmod$ operator for integers). Each operator takes exactly two integer operands. Every element in a list except for the last one is followed by a single space, and no other spaces are allowed anywhere in the list.

The input will contain 10 syntactically correct arithmetic expressions written in Steam, one on each line. To keep things simpler, only the integers 0 through 9 will be used as operands. Your job is to evaluate each expression and output the result. The result could be any positive or negative integer in the range $-32\,768$ to $32\,767$.

Sample Input

```
(+ 5 3)
(* 7 2)
(- 0 9)
(q 8 5)
(r 8 5)
(* (- 5 2) (+ 2 3))
(r (q 9 4) (* (+ (q 2 3) (- 5 (- 4 (* 2 3))))) 2))
(* 4 (* 4 (* 4 (* 4 (* 4 (* 4 4))))))
(* (* (* (* (* (* 4 4) 4) 4) 4) 4) 4)
(+ (q (- 9 3) (r 7 4)) (- (* 9 9) (* 0 1)))
```

Sample Output

```
8
14
-9
1
3
15
2
16384
16384
83
```