11193 Infinix

Infinix is a powerful processor that can evaluate any number of instructions (simple addition, subtraction, multiplication and division) at the same time (of course, the operands need to be available). To take the benefit of Infinix, a separate compiler is also required, as traditional compilers may produce codes that are not so suitable for Infinix. For example, to evaluate an expression

1+2+3+4 traditional compilers may produce code like

[I1] Add R1, 1, 2
[I2] Add R2, R1, 3
[I3] Add R3, R2, 4

Here $R3$ holds the summation. The problem with this code is that it introduces unnecessary data dependences (I2 cannot be executed until I1 is finished), which are difficult to overcome by hardware techniques. If addition operation takes 1 cycle in Infinix, for this code it will take 3 cycles. An intelligent compiler can produce another code for this expression

[I1] Add R1, 1, 2
[I2] Add R2, 3, 4
[I3] Add R3, R1, R2

For the above code 2 cycles will be enough to find the summation in $R3$. The difference is I1, I2 can be executed in parallel now.

Researchers are designing a new compiler for Infinix, which they call Infinicx. Infinicx always produces code, which can be evaluated in minimum time by Infinix. Though, Infinicx can produce code to its own, it follows the programmer assigned order and program semantics. For example, if the above expression is written by the programmer like 1+(2+3)+4, then Infinicx will produce the following code

Add R1, 2, 3
Add R2, 1, R1
Add R3, 4, R2

And it will take 3 cycles again. It cannot be converted to a code, which can be executed in 2 cycles. Also, Infinicx never changes any operators or operands or order of operands.

In this problem, you will be given the latency for add, sub, mul and div operations for Infinix. You need to find the minimum cycles necessary to evaluate the expression by Infinix.

Input

The input file will start with 4 integers a, s, m, d which are the number of cycles necessary for executing an addition, subtraction, multiplication and division operation in Infinix. The next line contains the expression (syntactically correct), which only contains digits, operators (`+-*/`) and parentheses. The expression length will not be greater than 500. There can be many such datasets. Input will be terminated by a line containing four zeroes.

Output

For each input, print in a single line the minimum number of cycles necessary to evaluate the expression, as described above.

Sample Input

```
1 2 3 4
1+2+3+4
1 2 3 4
1+(2+3)+4
1 2 3 4
1-2-3-4*1
1 2 3 4
1-2-3-4
0 0 0 0
```

Sample Output

```
2
3
6
6
```