F: Parenthesized Triplets

A student (not in ACM) is learning the language Lisp. He recognizes that arithmetic expressions are triples, but has not grasped the fact that the operator must come first. He places them randomly inside the nested parenthesized triples! Your task is to transform these expressions into correct Lisp, and evaluate it.

Input

Input may consist of multiple cases. Each case is presented on a single line. It will consist of properly and fully parenthesized nested triplets consisting of some ordering of two operands (positive integers or a nested triplet) and an operator (+, -, *, /) representing integer operations. Nesting of parentheses will be no more than 100 deep. The last case is followed by a line consisting of (+ 0 0) which is not to be processed. White space may appear anywhere except inside an integer, and is used to delimit pairs of integers (as shown in the sample input). The input will be such that no operation will cause an overflow or divide by 0 when using standard 32 bit signed integers.

Output

For each case, display two lines, the first with the case number and the calculated value of the expresssion, formatted as in the sample, with single spaces used as delimiters. The second line is a corrected Lisp expression. Single spaces should appear only between elements of each triplet, as shown below.

Sample Input

```
(1 6 -)
((3 + 4) 2 -)
(+ (4 - (2 5 *)) 6)
(+ 0 0)
```

Sample Output

```
Case 1: -5
(- 1 6)
Case 2: 5
(- (+ 3 4) 2)
Case 3: 0
(+ (- 4 (* 2 5)) 6)
```