UVa Online Judge

1662 Brackets Removal

Let us consider arithmetic expressions that consist of variables denoted by lower-case letters "a" to "z"; four *binary* arithmetic operations: addition ("+"), subtraction ("-"), multiplication ("*"), and division ("/"); opening ("(") and closing (")") round brackets. The normal order of precedence is used — multiplication and division have the highest precedence, addition and subtraction have the lowest precedence. Operations of the same precedence are evaluated from left to right (for example $a - b + c = (a - b) + c$).

Thus, the grammar for the expressions is the following:

$$\begin{aligned} < \text{expression} > &\longrightarrow < \text{term} > \mid < \text{expression} > + < \text{term} > \mid < \text{expression} > - < \text{term} > \\ < \text{term} > &\longrightarrow < \text{factor} > \mid < \text{term} > * < \text{factor} > \mid < \text{term} > / < \text{factor} > \\ < \text{factor} > &\longrightarrow < \text{variable} > \mid (< \text{expression} >) \\ < \text{variable} > &\longrightarrow a \mid b \mid \ldots \mid z \end{aligned}$$

Your task is to rewrite the given expression so that its semantics is not changed, but the resulting expression has the minimal number of round brackets.

You can remove any excessive brackets that do not change the order of evaluation, for example

$$\begin{aligned} (a + b) + (c) &\Rightarrow a + b + c, \\ (a * b)/(c) &\Rightarrow a * b/c, \end{aligned}$$

and you can rewrite expressions using the following rules:

- If A and B are arbitrary expressions, you can change $A + (B)$ to $A + B$, for example

$$a - g/h + (b + c - d + e * (f + h - i)) \Rightarrow a - g/h + b + c - d + e * (f + h - i).$$

- If A and B are arbitrary expressions, you can change $A - (B)$ to $A - B'$ where B' is obtained from B by replacing all top-level "+" operations to "-" operations and vice versa, for example

$$a - g/h - (b + c - d + e * (f + h - i)) \Rightarrow a - g/h - b - c + d - e * (f + h - i)).$$

- If A and B are arbitrary terms, you can change $A * (B)$ to $A * B$, for example

$$x/(y + z) * (a * (b - c)/d/(e/f)) \Rightarrow x/(y + z) * a * (b - c)/d/(e/f).$$

- If A and B are arbitrary terms, you can change $A/(B)$ to A/B', where B' is obtained from B by replacing all top-level "*" operations to "/" operations and vice versa, for example

$$x/(y + z)/(a * (b - c)/d/(e/f)) \Rightarrow x/(y + z)/a/(b - c) * d * (e/f).$$

You can think about these transformations as ones that only use "+" and "*" associativity, the fact that "-" is the reverse operation to "+", "/" is the reverse operation to "*", and nothing else.

You can apply the described transformations and remove excessive brackets as many times as you need to get the expression with the minimal number of round brackets.

Input

The input file contains several test cases, and each of them consists of a single line with the expression. Expression does not have any leading, trailing, or inner spaces and consists of at most 1000 characters.

Output

For each input case, write to the output file a single line with the same expression that is rewritten with the minimal number of round brackets. Do not write any spaces.

Sample Input

```
((a-b)-(c-d)-(z*z*g/f)/(p*(t))*((y-u)))
```

Sample Output

```
a-b-c+d-z*z*g/f/p/t*(y-u)
```