Problem A: Parenthesis

Source: `parenthesis.{c,cpp,java}`
Input: `console {stdin,cin,System.in}`
Output: `console {stdout,cout,System.out}`

To a computer, there is no difference between the expression **`(((x)+(y))(t))`** and **`(x+y)t`**; but, to a human, the latter is easier to read. When writing automatically generated expressions that a human may have to read, it is useful to minimize the number of parentheses in an expression. We assume expressions consist of only two operations: addition (+) and multiplication (juxtaposition), and these operations act on single lower-case letter variables only. Specifically, here is the grammar for an expression **`E`**:

```
E : P | P '+' E
P : F | F  P
F : V | '(' E ')'
V : 'a' | 'b' | .. | 'z'
```

The addition (**`+`**, as in **`x+y`**) and multiplication (juxtaposition, as in **`xy`**) operators are associative: **`x+(y+z)=(x+y)+z=x+y+z`** and **`x(yz)=(xy)z=xyz`**. Commutativity and distributivity of these operations should not be assumed. Parentheses have the highest precedence, followed by multiplication and then addition.

Input

The input consists of a number of cases. Each case is given by one line that satisfies the grammar above. Each expression is at most 1000 characters long.

Output

For each case, print on one line the same expression with all unnecessary parentheses removed.

Sample input

```
x
(x+(y+z))
(x+(yz))
(x+y(x+t))
x+y+xt
```

Sample Output

```
x
x+y+z
x+yz
x+y(x+t)
x+y+xt
```