Brackets

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

While working on the very important part of the code, the programmer Jonh used the autocorrecting tool to remove all extra brackets from the expression. Unfortunately, the tool accidentally removed all brackets from the line of code that John was not going to fix, and now the program cannot be compiled.

After John restored the original version, a new task came to his mind. Consider the following simplified version of a programming language:

- variables are single lowercase English letters: `var = ['a'-'z']`;
- pre-increment: `preincr = ++var`;
- post-increment: `postincr = var++ | preincr++`;
- operand: `operand = var | preincr | postincr | (operand)`;
- expression: `expr = operand | expr + expr`.

The character `'|'` is used to separate the options in the description above and **does not** appear in the expression.

John wants to know if there exists an algorithm that, for any such expression without the brackets, builds the expression with brackets in such a way that there is no ambiguity in parsing the expression. Specifically, the resulting expression must be valid, and for each `'+'` sign, we should be able to uniquely determine whether it is an addition sign, pre-increment or post-increment, and to which variable each increment is applied. Help him to find such a way to add brackets.

Input

The first line of the input contains the non-empty string that consists of the letters between '`a`' and '`z`' inclusively and the '`+`' signs, such as there are no two consecutive letters (i.e. all the variable names are of length 1). The length of the string goes not exceed 10^6.

You may assume that the string is a correct expression written in the programming language described in the statement.

Output

In a single line print the expression that is built from the given one by adding some brackets, such that all the pre-increments and post-increments included in the expression are uniquely defined. The expression must be correct in terms of the programming language described in the statement.

The number of pairs of the brackets must not be greater than the number of the variables (i.e. the number of the English letters in the input).

If there are more than one solutions, print any of them.

Examples

standard input	standard output
x+++y	(x++)+(y)
q+u+++h+++++o+q++	(q)+(u++)+(h++)+(++o)+(q++)