Computer scientists have a strange way of looking at arithmetic expressions. What a regular person sees as 3+4\*5 looks like + 3 \* 4 5 to a computer scientist. The former is called the infix notation, while the latter is called the prefix notation, or the Polish notation. In the Polish notation, the operator (e.g. +) comes first, followed by the left-hand side sub-expression, followed by the right-hand side subexpression. Each of the subexpressions again is in the prefix notation. One benefit is that no brackets are needed, they can be inferred: + 3 \* 4 5 can be unambiguously understood as + 3(\*45), that is, as 3+(4\*5).

Your task is to write a converter from prefix notation to infix notation for expressions that involve numbers in the range 0-9 (that is, every number is a single digit, there won’t be numbers such as 123), as well as two binary operators: + and \*. The converter should preserve the left-right order, that is, + 3 4 is translated to 3 + 4 and not to 4 + 3. To simplify things, every binary operation should be enclosed in brackets: instead of 3+4\*5 you should print (3+(4\*5)).

Input

A string of length up to 100 characters representing an expression in prefix notation. The string will contain only the following characters: 0123456789+\* and each character will be separated from the next by a single white space.

Output

A string representing the expression in infix notation. Do not use any whitespaces.

Sample Input

+ \* 3 + + 4 5 6 7

Sample Output

((3\*((4+5)+6))+7)