Baby Bob's Bracket Sequence (`brackets`)

Baby Bob is learning about mathematical expressions. He despises operands and operators, and only likes round brackets.

Figure 1: Baby Bob while learning.

He's got a sequence A of positive integers $A_1, A_2, \ldots, A_N$. He wants to *bracketize* the sequence. A *bracketized* sequence created from A is a sequence of strings $B_1, B_2, \ldots, B_N$ such that each B_i has length A_i, and B_i consists only of either opening brackets "`(`", or closing brackets "`)`", but not both.

For example let $A = (1, 3, 4)$.

- A possible *bracketized* sequence created from A is "`)`", "`)))`", "`((((`".
- The sequence "`)`", "`)()`", "`((((`" is not a *bracketized* sequence created from A, because the second element consists of both opening and closing brackets.
- The sequence "`(`", "`))))`", "`((((`" is not a *bracketized* sequence created from A, because the length of the second element is not 3.
- The sequence "`(`", "`)`" is not a *bracketized* sequence created from A, because it consists of only 2 strings.

Take the string $B_1 + B_2 + \ldots + B_N$ (i.e., concatenate the elements of the *bracketized* sequence). Bob wonders whether he can *bracketize* A so that the resulting string is a valid bracket sequence. A bracket sequence is valid if "`1`" and "`+`" characters can be inserted into it so that it becomes a valid mathematical expression. For example, "`((((()))))`" is a valid bracket sequence if $A = (1, 3, 4)$.

Write a program that finds such a bracket sequence or determines that it's impossible!

> ☞ Among the attachments of this task you may find a template file `brackets.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers A_i.

Output

You need to print a valid bracket sequence created from A or -1 if it's not possible to create one.

If there are multiple correct bracket sequences, output any.

Constraints

- $1 \leq N \leq 500$.
- $1 \leq A_i$ for each $i = 0 \ldots N-1$.
- $A_1 + A_2 + \ldots + A_N \leq 50\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (35 points) $N \leq 2$

– **Subtask 3** (35 points) $N \leq 20$ and $A_1 + A_2 + \ldots + A_N \leq 200$.

– **Subtask 4** (30 points) No additional limitations.

Examples

input	output
3 1 3 4	(((())))
4 2 2 1 1	(())()
2 2 1	-1

Explanation

The **first sample case** is explained in the statement.

In the **second sample case** the bracketized sequence is "((", "))", "(", ")".