F: Pen

Memory limit **128 MB**

Johnny loves order and is very upset by untidy parentheses. Lately he saw a string `){([`, which annoyed him since it was not a correct bracket sequence. Fortunately, Johnny had a pen in his pocket and he could add `(` at the beginning and `)}` at the end. Then he could sleep soundly again, since the string `(){([])}` is a correct bracket sequence.

Johnny took on repairing bracket sequences to be his life's mission. He will do this by adding new parentheses at the beginning and at the end of sequences he finds, so as to make them correct. Since his pen is running out of ink, Johnny would like to add the least possible number of brackets which will accomplish this. Help him! Write a program which will read a bracket sequence and either compute a shortest possible corrected bracket sequence for it and write it to standard output, or say that there is no way to correct it.

Correct bracket sequences are defined recursively as follows:

- the empty sequence is a correct bracket sequence,
- if S and T are correct bracket sequences, then their concatenation ST is also a correct bracket sequence,
- if S is a correct bracket sequence, then `(`S`)`, `[`S`]` and `{`S`}` are all also correct bracket sequences.

Input

The first and only line of input contains a nonempty sequence consisting of characters `(`, `)`, `[`, `]`, `{`, `}` and having length at most 1 000 000; this is the bracket sequence that Johnny wants corrected.

Output

The output should contain a corrected input sequence. If there are many possible ways to correct the input sequence that have the same minimum length, output any of them.

If it is impossible to correct the input sequence, output the word `NIE`.

Example

Input	Output
`){([`	`(){([])}`

This example is the one from the problem statement.

Input	Output
`([)}`	`NIE`

This sequence cannot be repaired.