Brackets

We are going to consider sequences consisting of three types of brackets: round `()`, square `[]`, and curly `{}`.

We say that a sequence is *proper* if every opening bracket has a matching closing bracket and vice versa, and pairs of matching brackets do not intersect. Formally, we define proper sequences recursively: The empty sequence is proper. If `A` is proper, then `(A)`, `[A]`, and `{A}` are proper. Finally, if `A` and `B` are proper, then `AB` is proper.

Given a sequence of brackets, decide whether it is proper, and if it is not, whether it can be turned into a proper one by appending at the end some number of brackets. In the later case, find a shortest sequence of brackets that can be appended.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 120$). The descriptions of the test cases follow.

Each test case consist of a single line containing a sequence of brackets, of length between 1 and 200 000.

The total length of sequences in all test cases does not exceed 2 500 000.

Output

For each test case output in a separate line:

- `YES`, if the given sequence is proper;
- `NO`, if it is not proper and it cannot be extended to a proper sequence;
- otherwise, a shortest sequence of brackets that can be appended at the end of the given sequence in order to turn it into a proper sequence.

Sample input

```
4
{}[]
(((
()]
([
```

Sample output

```
YES
)))
NO
])
```