551 Nesting a Bunch of Brackets

In this problem we consider expressions containing brackets that are properly nested. These expressions are obtained by juxtaposition of properly netsted expressions in a pair of matching brackets, the left one an opening and the right one a closing bracket.

`( a + $ ( b = ) ( a ) )` is properly nested

`( a + $ ) b = ) ( a ( )` is not

In this problem we have several pairs of brackets, so we have to impose a second condition on the expression: the matching brackets should be of the same kind. Consequently `(())` is OK, but `([))` is not. The pairs of brackets are:

```
(   )
[   ]
{   }
<   >
(*   *)
```

The two characters '`(*`' should be interpreted as one symbol, not as an opening bracket '(' followed immediately by an asterisk, and similarly for '`*)`'. The combination '`(*)`' should be interpreted as '`(*`' followed by '`)`'.

Write a program that checks wheter expressions are properly nested. If the expression is not properly nested your program should determine the position of the offending bracket, that is the length of the shortest prefix of the expression that can not be extended to a properly nested expression. Don't forget '`(*`' counts as one, as does '`*)`'. The characters that are not brackets also count as one.

Input

The input is a text-file. Each line contains an expression to be checked followed by and end-of-line marker. No line contains more than 3000 characters. The input ends with a standard end-of-file marker.

Output

The output is a textfile. Each line contains the result of the check of the corresponding inputline, that is `YES` (in upper case), if the expression is OK, and (if it is not OK) `NO` followed by a space and the position of the error.

Sample Input

```
(*a++(*)
(*a{+}*)
```

Sample Output

```
NO 6
YES
```