Compilers and Parsers

Lira is now very keen on compiler development. She knows that one of the most important components of a compiler, is its parser.

A parser is, in simple terms, a software component that processes text, and checks it's semantic correctness, or, if you prefer, if the text is properly built.

As an example, in declaring and initializing an integer, in C/C++, you can't do something like:

```
int = x ;4
```

as the semantics of such statement is incorrect, as we all know that the datatype must precede an identifier and only afterwards should come the equal sign and the initialization value, so, the corrected statement should be:

```
int x = 4;
```

Today, Lira is concerned with an abstract instruction which is composed of the characters "<" and ">" , which she will use on the design of her language, L++.

She is using it as an abstraction for generating XML code Tags in an easier fashion and she understood that, for an expression to be valid, a "<" symbol must always have a corresponding ">" character somewhere (not necessary immediately) after it. Moreover, each ">" symbol should correspond to exactly one "<" symbol.

So, for instance, the instructions:

```
<<>>


<>


<><>
```

are all valid. While:

```
>>


><><
```

are not.

Given some expressions which represent some instructions to be analyzed by Lira's compiler, you should tell the length of the longest prefix of each of these expressions that is valid, or 00 if there's no such a prefix.

Input Format

Input will consist of an integer TT denoting the number of test cases to follow.

Then, TT strings follow, each on a single line, representing a possible expression in L++.

- $1 \le T \le 500 1 \le T \le 500$
- $1 \le 1 \le$ The length of a single expression $\le 10^6 \le 10^6$
- The total size all the input expressions is no more than $5 \cdot 10^6 5 \cdot 10^6$

Output Format

For each expression you should output the length of the longest prefix that is valid or 00 if there's no such a prefix.

Sample test

inputcopy
3 <<>> >< <>>>
outputcopy
4 0 2