Source: **CodeChef |** Problem Code: **COMPILER**

Problem Statement:
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 process 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 data-type must precede an identifier and only afterwards should come the equals 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 character "<" and ">" which she will use on the design of her language, L++ :D

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 ">" should correspond to exactly one "<" symbol.
So, for instance, the instructions:

<<>>

<>

<><>

are all valid

>>

><><

are not.
Given some expression which represent some instructions to be analysed by Lira's compiler, you should take the expression or the symbols as input and tell the length of the longest prefix of each of these expression that is valid , or 0 if there's no such prefix. In addition to that just print something to show whether the given expression is valid or not .

EXAMPLE:
Input:
<<>>
Output:
largest_prefix: 4
Expression is valid.

Input:
><
Output:
largest_prefix: 0
Expression is invalid.

Input:
<>>>
Output:
largest_prefix: 2
Expression is invalid.