3429 Pasting Strings

You are part of a team implementing an HTML editor and have been tasked with the problem of implementing the cut/copy/paste functions. One of your goals is to preserve the formatting of selected text, even though that formatting may be determined by HTML tags outside the range of the actual selected text.

You will be provided with a block of formatted text, a starting position B, and an ending position E. Your program should output the text of the substring of that text from B (inclusive) to E (exclusive), prepending and appending formatting tags as necessary so that the output is well formed and has the same format as it had in its original position.

For the purposes of this problem, format tags consist of an opening tag (such as "`<b>`"), followed by some text, followed by a closing tag (such as "`</b>`"). Opening and closing tags are paired ("`</`*whatever*`>`" closes "`<`*whatever*`>`") and are considered opened between the opening tag and the closing tag.

A tag may not be closed unless it is the most recent unclosed tag (e.g., "`<i>abc<b>def</i>ghi</b>`" is illegal). A tag may not be opened if it is already open (e.g., "`<b><b>recursive b</b></b>`" is illegal).

Input

Input data will consist of multiple test cases. Each test case will consist of one line of input of the form

B E $TEXT$

where B is an integer giving the (inclusive) beginning location of the substring, E is an integer giving the (exclusive) ending location of the substring, and $TEXT$ contains the text from which to extract the substring. The $TEXT$ begins after a single blank character immediately following E, and continues to the end of the line. B and E will be specified so that $0 \leq B \leq E \leq length(TEXT)$.

End of input will be signaled by the line '`-1 -1 `' with a single space following the second '`-1`'.

No input line will be longer than 200 characters.

The $TEXT$ will be composed of characters with an ASCII value ≥ 32 (the ASCII space) and ≤ 126 (the ASCII '~'). Opening tags will be of the form '$<X>$' where X contains at least 1 character and is composed entirely of the characters '`a`' to '`z`', '`A`' to '`Z`', '`0`' to '`9`', and '`-`'. Closing tags will be of the form '`</`X`>`'. The character '`<`' will only occur in the input as the first character of an opening or closing tag.

The input text will be well formed — all opening tags will be matched with a closing tag, all closing tags will match an opening tag, each closing tag will close the most recent unclosed tag, and tags will not be recursive (each tag must be closed prior to reopening).

$0 \leq B \leq E \leq length(TEXT)$. Neither B nor E will reference a character that is part of an opening or closing tag except for the character '`<`'.

Output

For each test case your program should print a single line containing the substring of $TEXT$ from B (inclusive) to E (exclusive), prepending the substring with opening tags and appending the substring with closing tags as necessary so that the output line is well formed and has the same set of open tags as when it was included in the original $TEXT$.

Sample Input

```
0 15 Testing<b>!</b>
18 23 <big>100, <bigger>1000, <biggest>10000</biggest></bigger></big>
4 4 <b>123</b>
0 16 :-/ :-> :-) :-<-> </->
-1 -1
```

Sample Output

```
Testing<b>!</b>
<big><bigger>1000,</bigger></big>
<b></b>
  :-/ :-> :-) :-
```