Exactly XML

input file: `xml.in` output file: `xml.out`

Extensible Markup Language (XML) is a new text formatting language. It consists of a number of different elements. An element is marked by opening and closing tags. Tags are text inside less than and greater than signs ('<' and '>'). A closing tag is identical to an opening tag except after the '<' there is a slash ('/'). For example, to indicate text that should be emphasized the following line might be used:

```
<em> very important point </em>
```

Elements may be nested, so to indicate a line should be emphasized and underlined, either of the following may be used:

```
<underline><em> very important point </em></underline>
<em><underline> very important point </underline></em>
```

Notice that in nesting, tags must be matched, so a closing tag must match the most recently opened tag. The following line is not valid XML because the `</underline>` tag does not match the most recently opened tag, the `<em>` tag:

```
<underline><em> very important point </underline></em>
```

XML allows tags to be self closing. A self-closing tag is an opening tag that does not need a closing tag. A tag is made to be self-closing by putting a '/' before the '>' in the tag. For example, the following might be used to put a smiley face in a document:

```
<smileyface/>
```

It is impossible to nest anything in a self-closing tag. There are other options for XML documents, but for this assignment, only tags and enclosed text are allowed.

Write a program to parse valid XML documents and print the resulting parse tree.

Input

The input to your program will be a valid XML document. A valid XML document consists of text and tags. Text may contain any characters except the less than sign ('<'). A tag begins with a '<'. If the tag is a closing tag, it will have a '/' immediately after the '<'. After the '<' (and '/', if there is one) there will be the name of the tag. The name of the tag will be a string of up to 20 letters and digits and the underscore('_'). Case is significant in the name of a tag, so `<em>` and `<EM>` are different tags. If the tag is a self-closing tag, it will have a '/' after the name. After the name (and '/', if there is one), the tag ends with a '>'. There will be no blanks in tags.

Your program may assume the tags in the document are appropriately nested.

Output

The output of your program should be a list of the tags read by your program. Each tag must be written on its own line, in the same order as it appears in the input. They should be indented to indicate nesting. Tags which are not within other tags (level-0 tags) should be written beginning in the first column of the line. Tags which are nested inside just level-1 tags (level-2 tags) should be written after three periods. Tags which are nested inside just level-2 tags (level-3 tags) should be written after six periods. In general, level-*n* tags should be written after (*n*-1)*3 periods. There should be no space between the periods or between the periods and the tags. The tags should be written exactly as they appear in the input.

Sample Input

```
<recipe>
<title>Spicy Gator <smileyface/>Bites</title>
<source>Georgia <red>Bulldog</red>
Cookbook</source>

<ingredient_list>
<item>1 Gator Tail
   <preparation>Diced into one inch cubes.
   </preparation></item>
<item>Hot grease <danger_indication/></item>
</ingredient_list>

<preparation><step>Drop meat into oil.</step>
<step>Remove meat from oil.</step>
</preparation>
</recipe>
```

Sample Output (corresponding to sample input)

```
<recipe>
...<title>
......<smileyface/>
...</title>
...<source>
......<red>
......</red>
...</source>
...<ingredient_list>
......<item>
.........<preparation>
.........</preparation>
......</item>
......<item>
.........<danger_indication/>
......</item>
...</ingredient_list>
...<preparation>
......<step>
......</step>
......<step>
......</step>
...</preparation>
</recipe>
```