25th Annual ACM International Collegiate Programming Contest

2000 Greater New York Regional

Problem E – A Well-Formed Problem

Background

XML, eXtensible Markup Language, is poised to become the *lingua franca* of structured data communication for the foreseeable future, due in part to its strict formatting requirements. XML parsers must report anything that violates the rules of a *well-formed* XML document. An XML document is said to be well-formed if it meets all of the well-formedness constraints as defined by the World Wide Web Consortium (W3C) XML specification.

XML documents are composed of units called elements, that contain either character data and/or other elements. Elements may also contain within their declaration values called attributes. Consider the following XML document:

```
<?xml version="1.0"?>
<customer>
        <name>
                <first>John</first>
                <last>Doe</last>
        </name>
        <address>
                <street>
                        <number>15</number>
                        <direction>West</direction>
                        <name>34th</name>
                </street>
                <city>New York</city>
                <state-code>NY</state-code>
                <zip-code format="PLUS4">10001-0001</zip-code>
                <country-code>USA</country-code>
        </address>
        <orders/>
</customer>
```

The bold identifiers contained within angle brackets are the elements of the document. The italicized identifier "format" within the "zip-code" element is an attribute of that element. All elements, with the exception of "orders", have a start and an end declaration, also called a tags. The "orders" element is an empty element, as indicated by the "/>" sequence that closes the element, and does not require a separate end-tag. The first line is a processing instruction for an XML parser and is *not* considered an element of the document.

The rules for a well-formed document are:

1. **There is exactly one element that is not contained within any other element.** This element is identified as the "root" or "document" element. In the example above, "customer" is the document element.
2. **The structure of an XML document must nest properly.** An element's start-tag must be paired with a closing end-tag if it is a non-empty element.
3. **The name in an element's end-tag must match the element type in the start-tag.** For example, an element opened with `<address>` must be closed by `</address>`.
4. **No attribute may appear more than once in the same start-tag or empty-element tag.**
5. **A parsed element must not contain a recursive reference to itself.** For example, it is improper to include another address element within an address element.
6. **A named attribute must have an associated value.**

Input

The input file will contain a series of XML documents. The start of each document is identified by a line containing only the processing instruction “`<?xml version=”1.0”?>`”. The end of the input is identified by a line containing only the text “`<?end?>`” (this is not a true XML processing instruction, just a sentinel used to mark the end of the input for this problem). As with all XML documents, white space between elements and attributes should be ignored. You may make the following assuptions with regard to the input.

- The only processing instruction that will be present is the XML version processing instruction, and it will always appear only at the beginning of each document in the input.
- Element and attribute names are case-sensitive. For example, <Address> and <address> are considered to be different.
- Element and attribute names will use only alpha-numeric characters and the dash “-” character.
- XML comments will not appear in the input.
- Values for attributes will always be properly enclosed in double quotes.

Output

For each input XML document, output a line containing the text “well-formed” if the document is well-formed, “non well-formed” otherwise.

Example

Input

```
<?xml version=”1.0”?>
<acm-contest-problem>
        <title>A Well-Formed Problem</title>
        <text>XML, eXtensible Markup Language, is poised to become the lingua franca of
structured data communication for the foreseeable future. [...]</text>
        <input>probleme.in</input>
        <output>probleme.out</output>
</acm-contest-problem>
<?xml version=”1.0”?>
<shopping-list>
        <items>
                <item quantity=”1” quantity=”1”>Gallon of milk</item>
                <item>Frozen pizza
        </items>
</Shopping-list>
<errand-list>
        <errand>Get some cash at the ATM
                <errand>Pick up dry cleaning</errand>
        </errand>
</errand-list>
<?end?>
```

Output

```
well-formed
non well-formed
```