CS 2204: Final Exam

Assigned: December 2, 2005
Date Due: December 13, 2005, 11:59am

---

1. Write a `bash` script called `validator` that takes an XML document as input and determines whether the document is well-formed (we will define below what this means).

   An XML document looks exactly like a HTML document except that the tag names can be arbitrary. For instance, here is a simple XML document that uses tags such as `breakfast`, `eat`, and so on. Recall that opening tags are given by the tag name enclosed in angle brackets, whereas closing tags have an extra `/` character after the beginning angle bracket.

   ```
   <breakfast>
   <eat>bagel</eat>
   <drink>coffee</drink>
   <eat><action>toasted</action> bread slice</eat>
   <drink>coffee <action>with cream and sugar</drink></action>
   <eat>scrambled eggs</Eat>
   <eat how="greedily">hash browns</eat>
   <eat how=slowly>grits</eat>
   <eat>waffles
   </breakfast>
   ```

   We can think of an XML document as having a hierarchical structure just as a directory tree; for instance, the `action` tag in the fourth line above is a child of `eat` (also on the fourth line), which is a child of `breakfast` (on the first line). To be well-formed, an XML document must obey some rules:

   (a) There must be a single root, i.e., all the rest of the document must be contained inside one type of tag. The above document satisfies this rule, since the single root involves the `breakfast` tag.

   (b) All tags must be closed and case does indeed matter. In the case of the `breakfast` tag, we see a matching closing tag on the last line. However, in the sixth line, there appears to be a mismatch of case with the `eat` tag. Similarly, in the line about `waffles`, the closing tag appears to be missing. Therefore the document is not well-formed according to this rule.

   (c) Attribute values must be quoted using double quotes. Attributes are (name, value) pair assignments that are made within the angle brackets in the body of the opening tag. For instance, the line about `greedily` eating `hash browns` does indeed have the attribute value quoted when being assigned to the variable called `how`. However, the line about `slowly` eating `hash browns` violates this rule. Therefore the document is not well-formed according to this rule. There can be any number of attribute assignments inside the opening tag, e.g.,

   ```
   <eat how="greedily" using="fork" using="napkin">hash browns</eat>
   ```

There cannot be attribute assignments in a closing tag.

(d) This is probably the most important rule: tags must not overlap. For instance, in the line about `toasted bread slice`, note that the `action` part begins after `eat` and ends before `eat` ends. This is a legal usage. However, the next line, involving `with cream and sugar`, violates this rule because the opening and closing tags of `action` and `drink` overlap.

You must conclude that a document is not well-formed if even one of the above rules is violated. In coding this assignment, keep in mind that XML documents are free-flowing, so closing tags can be found on a different line than the opening tag (e.g., the `breakfast` tag above). However, you can assume that tag and attribute names will not be split up across lines, so that:

```
<eat how="greedily"
   using="fork"
using="napkin">hash
browns</eat>
```

is a possible (legal) input file, but

```
<ea
t how="greedily"
   using="fork"
using="napkin">hash
browns</eat>
```

is not and should be flagged as an ill-formed document. You may also assume that the angle brackets `<` and `>` will be used only for tag opening and closing, and not as content inside a tag. Futhermore, you can assume that there will be no space after/before the angle brackets, but there could be spaces in the content of the tag. So,

```
<drink>cof fee</drink>
```

is legal, but

```
< drink>coffee</drink>
```

is not.

Your script will be invoked as:

```
validator test1.xml
```

where `test1.xml` is an example input XML file. Depending on the result of your analysis, you must print one of the following two messages:

```
test1.xml is well-formed.
test1.xml is ill-formed.
```

You can choose to terminate your script as soon as you find a violation of the rules. You are not allowed to call/use any libraries available for parsing HTML/XML files and must develop all the text extraction code yourself.

Extra Credit:

1. (10%): Allow for empty XML elements. For instance, instead of writing

   ```
   <eat></eat>
   ```

   which is legal according to our above rules, we can instead write

   ```
   <eat/>
   ```

   Extend your program to allow for such elements.

2. (20%): Given an (optional) additional argument to `validator` using the `-u` option, such as:

   ```
   validator test1.xml -u eat
   ```

   your shell script need only check if that portion of the document inside `eat` tags is well-formed. This means that each of the `eat` usages must have matching open and closed tags and any XML text inside them is also well-formed, but you do not have to worry about XML text outside of `eat` tag pairs. If the tag given by the additional argument does not appear in the XML document at all, then you must conclude that, trivially, the document is well-formed.

3. (20%): Given an (optional) additional argument to `validator` using the `-i` option, such as:

   ```
   validator test1.xml -i 1
   ```

   this says that we can ignore (hence the option `-i`) rule 1 in the list of rules for checking. So we need only check if rules 2, 3, and 4 apply. You may assume that only rules 1 and 3 are 'ignorable,' and that attempts made to ignore rules 2 and 4 should not be accommodated.

   Keep in mind that the `-u` and `-i` tags can be used many times in a single invocation of validator, e.g.,

   ```
   validator test1.xml -i 1 -u eat -u drink -i 3
   ```

   means that we must apply only rules 2 and 4 and check only tags `eat` and `drink` in `test1.xml`.