Programming with Mutually Self-Referential Data

COMP 50

Fall 2013

This homework is due at 11:59PM on **Monday, October 21**. Submit your solutions in a single file using the COMP 50 Handin button on DrRacket; the homework is the `mutual-reference` homework.

All the exercises should be done using the Beginning Student Language *with list abbreviations*.

Alert

The point of the homework is to help you learn, by repetition, the full design recipe for mutually self-referential data. If you follow the design recipe, you will take some time over the first problems, blitz through problems 2 to 4, then take a little time over problem 5. If you don't follow the design recipe, you will get bogged down and you won't finish.

Data Description

X-Expressions

An X-expression models information that you would find in an XML or HTML document, such as a web page.

A *tag* is a symbol which is one of

- `'html`, `'header`, `'title`, `'body`, `'h1`, `'p`, `'ul`, `'ol`, `'li`, `'em`, `'b`, `'tt`, or other symbol defined by the World-Wide Web Consortium

An *element* is one of

- A string
- A list `(cons tag list-of-elements)`

A *list-of-elements* is one of

- `empty`
- `(cons element list-of-elements)`

Here is a data example for *element*

```
'(html (header (title "COMP 50 Home page"))
       (body (h1 "Welcome to COMP 50")
             (p "The following information is available:"
                 (ul (li "Lecture notes")
                     (li "Lecture and deadline schedule")
                     (li "Homework problems and solutions")))))
```

Finger exercises

From the first edition textbook, I am recommending these finger exercises:

- Exercises 15.1.2, 15.1.4, 16.2.1, 16.2.2, and 16.3.4

I am also recommending the following finger exercises:

i. *Define a function* that counts the number of strings in an X-expression element (including all nested elements).

ii. *Define a function* that returns a list containing every *tag* used in an X-expression element (including all nested elements). Even if a tag occurs more than once, make sure it occurs only once in the output list.

Problems to submit

Exploring how HTML is rendered by web browsers

1. In HTML, elements tagged with `li` are supposed to appear only inside *ordered* or *unordered* lists (tagged with `ul` and `ol`).

   Define a function that tells if anywhere in a given element, there are "bad" `li` elements that occur outside any ordered or unordered list element. As a couple of examples,

   - In the example element shown above, all three `li` elements occur safely inside the `ul` element. So that element has no bad `li` elements.
   - In the example element shown above, if `ul` were changed to `p`, all the `li` elements would be bad.

2. When `ul` lists are nested, `li` elements are tagged with different markers. Here is an example:

   - Level One unordered
     - Level Two unordered
       * Level Three unordered
         · Level Four unordered

   The printed version uses a bullet, a dash, a star, and a dot. Your web browser probably shows something different.

   Define a function that returns the *deepest* nesting depth of `ul` elements in a given X-expression element. This number should equal the number of markers needed to distinguish different levels of nesting. If the element contains no `ul` elements, the function should return zero, since no markers are needed.

   If `ul` and `ol` elements are nested within each other, your function should count only `ul` elements.

   BSL vocabulary: built-in BSL function `max` returns the largest of its numeric arguments.

3. When `ol` lists are nested, `li` elements are tagged with different kinds of numbering systems. Here is an example:

   a. Level Two ordered: alphabetical (the numbered problems are Level One)
      i. Level Three ordered: roman
      ii. Level Three again

   b. Level Two again
      i. Level Three restarts
      ii. Level Three again
         A. Level four ordered: capital alphabet

c. Level Two yet again

Define a function that returns the *deepest* nesting depth of `ol` elements in a given X-expression element. This number should equal the number of different kinds of numbering systems needed to distinguish different levels of nesting. If the element contains no `ol` elements, the function should return zero, since no numbered lists are needed.

If `ul` and `ol` elements are nested within each other, your function should count only `ol` elements.

4. Every kind of list, whether ordered or unordered, requires indentation to the right. *Define a function* that computes the maximum list indentation required by an X-expression element. This example, combined with the problem number, shows a maximum list indentation of four:

   - Level Two overall (unordered)
     a. Level Three overall (ordered)
        – Level four overall (unordered)

   This function should count both `ul` and `ol` elements

5. Read the next paragraph, so you will know where you are going, then proceed as follows: *analyze* the data "all possible BSL values" and break it down into choices that make sense for solving the problem. Then *write a data definition* for "all possible BSL values." The data definition may include the choice "any value other than those listed above." *Your data definition may need to be mutually self-referential with another data definition.*

   Define a function that takes any BSL value and returns a Boolean that says whether the value respects the data definition for *element*. Your function may assume that *any* symbol is a valid tag. If your data definition includes a choice "any value other than those listed above," then for that choice you may use `else`.

Karma Problems

A. *Define a function* that removes all tags from an XML element, leaving only a list of strings.

B. *Define a function* that removes all tags from an XML element *except* the `ol`, `ul`, and `li` tags. The result value should be a list of elements. *Elements whose tags are removed should be preserved*, so for example the element

```
'(html (header (title "COMP 50 Home page"))
       (body (h1 "Welcome to COMP 50")
             (p "The following information is available:"
                (ul (li "Lecture notes")
                    (li "Lecture and" (b "deadline") "schedule")
                    (li "Homework problems and solutions")))))
```

should be converted into something like this:

```
'("COMP 50 Home page"
  "Welcome to COMP 50"
  "The following information is available:"
  (ul (li "Lecture notes")
      (li "Lecture and" "deadline" "schedule")
      (li "Homework problems and solutions")))
```

In addition to the usual tests, test your function by confirming that removing the tags does not change the answers produced by the functions you wrote for problems 2 to 5.