CS 2500, Spring 2014
Problem Set 11

Due date: 7pm Tuesday April 1, 2014

Programming Language: Intermediate Student Language with Lambda

Purpose: to practice designing functions that consume two complex inputs.

You **must** follow the design recipe. The graders will look for data definitions, signatures, purpose statements, examples/tests, and properly organized function definitions. For the latter, you **must** design templates. You do not need to include the templates however. If you do, make sure to comment them out.

Problem 1. Exercise 318 from HtDP/2e

Problem 2. Exercise 319 from HtDP/2e

Problem 3. Exercise 320 from HtDP/2e

Problem 4. Exercise 326 from HtDP/2e

Problem 5. Exercise 327 from HtDP/2e

Problem 6. Exercise 329 from HtDP/2e

Problem 7. Exercise 326 from HtDP/2e

Problem 8. Here are two data definitions:

```
(define-struct leaf ())
(define-struct growth (next))
(define-struct fork (left right))

;; Tree is one of:
;; -- (make-leaf)
;; -- (make-growth Tree)
;; -- (make-fork Tree Tree)

;; Direction is one of:
;; -- 'left
;; -- 'right
```

a) Develop the template for a function that consumes a `Tree` and a `[List-of Direction]`.

b) Design the function `navigate`. It consumes a `Tree` and a [`List-of Direction`]. It navigates the tree at the specified places, that is, for a 'growth', it goes straight (because there is no choice). For a 'fork', it follows the next direction. If there are no more directions, it returns the sub-tree at that point. Decide what should happen if a leaf is reached but the `[List-of Direction]` is not empty.

Problem 9. Recall that an `S-expression` is defined as:

```
; An S-expr (S-expression) is one of:
; - Atom
; - SL
; An SL [List-of S-expr] is one of:
; - empty
; - (cons S-expr SL)
; An Atom is one of:
; - Number
; - String
; - Symbol
```

a) Develop the templates for a function that consumes two `S-expressions`

b) Design a program that will determine if two `S-expressions` contain the same atoms regardless of the ordering. For example:

```
(contains-same-atoms?  '(1 2 3 () ("r" b))
                       '("r" 1 (2) 3 b))
```

would return true.

Do *not* solve this by flattening the `S-expressions` into `[List-of Atom]` first.