CS 115 – Dubai – Fall 2012

Assignment 05

Due: Thursday, November 8, 2012 at 11:59 p.m.

- Do **not** use reverse or member in any of your solutions.
- You may want to include defined constants to help reduce the writing for the examples and tests.
- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions that include the design recipe, where appropriate.
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include references to the parameter names of your functions.
- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
- Do not send any code files by email to any course staff. It will not be accepted by course staff as an assignment submission. Course staff will not debug code emailed to them.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Read each question carefully for restrictions.
- Read the course Web page for more information on assignment policies and how to organize and submit your work. Follow the instructions in the style guide. Specifically, your solutions should be placed in files a05qY.rkt, where Y is a value from 1 to 3.
- Download the interface file from the course Web page.

Language level: Beginning Student with List Abbreviations
Coverage: Modules 5 and 6

Useful structure and data definitions:

```
(define-struct competitor (name result))
;; A competitor is a structure, (make-competitor n r), where
;;   n is a string, representing the name of the competitor
;;   r is a positive number, representing a measurement of the result in an event

(define-struct event (desc gold silver bronze))
;; An event is a structure, (make-event d g s b), where
;;   d is a symbol, describing the competitive event
;;   g is a competitor, representing the winner of the gold medal in the event
;;   s is a competitor, representing the winner of the silver medal in the event
;;   b is a competitor, representing the winner of the bronze medal in the event

;; An association list (al) is either
;;   empty or
;;   (cons (list k v) alst) where
;;      k is a number (the key)
;;      v is a string (the value)
;;      alst is an association list.
```

1. In the Olympics, sometimes athletes are trying to achieve more than their fellow competitors, and sometimes they are trying to achieve less than their fellow competitors. For example, in a race the goal is to get the shortest time, but in gymnastics the goal is to get the highest score. Write a function called `compute_medalists` consumes a symbol (`event-name`), a list of at least three competitor structures (`athletes`) and a `boolean` value (`less-is-better`) and produces an `event` structure. When the value of `less-is-better` is `true`, then the `event` will include the `competitor` structures from `athletes` representing the three competitors with the lowest values in their `result` field. When the value of `less-is-better` is `false`, then the `event` will include the `competitor` structures from `athletes` representing the three competitors with the highest values in their `result` field. For example:

Assignment 05

Due: Thursday, November 8, 2012 at 11:59 p.m.

```
(compute-medalists ('100-m-dash, (list (make-competitor "a" 10.5)
                                       (make-competitor "b" 11.1)
                                       (make-competitor "c" 9.8)
                                       (make-competitor "d" 10.1)
                                       (make-competitor "e" 10.7)
                                       (make-competitor "f" 9.7)) true)
```

produces

```
(make-event '100-m-dash
   (make-competitor "f" 9.7) (make-competitor "c" 9.8) (make-competitor "d" 10.1))
```

Notes:

- Module 6 included an example that showed how to sort a list of numbers using a technique called insertion sort. You may modify this code to use as helper functions for this question. However, if you do that, make sure you indicate that the original code came from the course notes.
- You may assume that there are no ties in the results of the competitors for a single event.

2. Write a function called `factors` that consumes a positive integer (`n`), and produces a list of positive factors of that number. Note that `1` and `n` are always included as factors of `n`. The list that is produced should be in ascending order. For example, `(factors 36)` should produce `(list 1 2 3 4 6 9 12 18 36)`.

3. Module 6 describes an association list (`al`) as a way of implementing a dictionary. Recall that association lists should have unique keys, but the values may be duplicated. Write the function `matching-values` that consumes an association list (`dictionary`) and a string (`s-value`) and produces list of keys all the key/value pairs in `dictionary` that have a value that matches `s-value`. If there is no key/value pair with a value that is equal to `s-value`, then the function should produce an empty list. In the list that is produced, the keys should be in the same relative order as they were in `dictionary`. For example

```
(matching-values (list (list 1 "a") (list 3 "b") (list 4 "c")
                       (list 8 "a") (list 10 "a") (list 12 "b")) "a")
```

produces

```
(list 1 8 10).
```