CS 115 – Dubai – Fall 2012

Assignment 03

Due: Thursday, October 11, 2012 at 11:59 p.m.

- 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 reference 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 a03qY.rkt, where Y is a value from 1 to 3.
- Download the interface file from the course Web page.

Language level: Beginning Student
Coverage: Module 4

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 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

(define-struct set-card (number colour shape shading))
;; A set-card is a structure (make-set-card n c sp sd) where
;;   n is a natural number from 1 to 3
;;   c is a symbol, one of 'red, 'green, or 'purple
;;   sp is a symbol, one of 'diamond, 'oval, or 'squiggle, and
;;   sd is a symbol, one of 'solid, 'striped, or 'open
```

1. There are many athletic events that take place during the Olympics. The top competitors are awarded medals: gold, silver, and bronze. After the event, the athletes are tested for banned substances. If the tests come back with a positive result, then that athlete is stripped of his/her medal, and everyone in the event who originally placed lower in the event is awarded one place higher. Write a function called `remove-gold` that consumes an `event` structure (`ev`) and an `athlete` structure (`fourth-place`) and produces a new `event` structure. The new `event` structure will have the old silver-medalist as the new gold medalist, the old bronze medalist as the new silver medalist, and the fourth-place athlete as the new bronze-medalist. There will be one other change in the new event. The new gold medalist will have three asterisks `"***"` at the beginning of the competitor's name.

Assignment 03

Due: Thursday, October 11, 2012 at 11:59 p.m.

For example,

```
(remove-gold   (make-event '100-m
                     (make-competitor "Ben Johnson" 9.79)
                     (make-competitor "Carl Lewis" 9.92)
                     (make-competitor "Lindford Christie" 9.97))
               (make-competitor "Calvin Smith" 9.99))
```

produces

```
(make-event '100-m   (make-competitor "***Carl Lewis" 9.92)
                     (make-competitor "Lindford Christie" 9.97))
                     (make-competitor "Calvin Smith" 9.99))
```

The following two questions are based on a game called Set©. Set© is a game with cards that have four features: a shape, a pattern, a colour, and a number of objects. For more information about the game see: http://en.wikipedia.org/wiki/Set_(game). Each card has one of three possible shapes: diamond, squiggle, and oval. Each card has one of three possible shadings: solid, striped, and open. Each card has one of three possible colours: red, green, or purple. Each card has one, two or three of the shapes on it. An example of a card description is: *three, green ovals with solid shading*. Each card in the game is unique. In the game of Set©, you must find combinations of three cards where each of the features on the three cards are either all the same or all different. For example: this is a set

- *three, green ovals with solid shading*
- *three, red ovals with striped shading*
- *three , purple ovals with open shading*

because all the cards have three ovals, but each card has a different colour and a different shading.
This is not a set

- *one green squiggle with solid shading*
- *two green squiggles with solid shading*
- *three green squiggles with striped shading*

because although all the cards have green squiggles and a different number of shapes, two of the three cards have the same shading.

2. Write a function called `valid-set?` that consumes three different `set-card` objects and produces `true` when they form a set, and `false` otherwise.

3. Write a function called `complete-set` that consumes two different `set-card` objects and produces a `set-card` that would satisfy the requirements such that the three cards together (the two cards that are consumed and the one card that is produced) form a set. For example, if the function consumes `(make-set-card 1 'purple 'diamond 'solid)` and `(make-set-card 3 'red 'oval 'striped)` then the only card that could complete a set with these two cards would be `(make-set-card 2 'green 'squiggle 'open)`.