CS 2500, Spring 2012
Problem Set 3

Due date: January 29, 2012 at midnight

For Problem Set 3 and later, homework is submitted via the automated homework server.
Note: Hardcopy submissions not accepted. Email submissions not accepted.

The goal of this problem set is to practice designing data representations using structures and functions for processing such forms of data. Furthermore, you MUST follow the design recipe and the guidelines when you develop functions.

Part 1:

HtDP Problems: 6.4.2, 6.4.3, 6.5.2

Part 2:

Here is a data definition for a traffic light:

```
(define-struct tlight (radius color))
;; Tlight = (make-tlight Number String)
;; Constraints:
;; The string is either “red” “yellow” or “green”
```

Use these definitions for the following exercises:

1. Design the function `change` from `Tlight` to `Tlight`. It changes the light from red to green to yellow to red again.
2. Design the function `draw-light`, which draws a circle from a tlight.
3. Optional: After you have developed these functions, you may wish to add the following lines to your program:

   ```
   (big-bang (make-tlight 60 “red”)
             (on-tick change 0.5)
             (to-draw draw-light))
   ```

   If you do, be sure to delete them before you turn in your solution. We will subtract points if you leave them in.

Part 3:

Develop a program that allows one to control the movement of a ball across the screen. The ball can move on straight lines only, that is, up, down, left, or right. Pressing the arrow keys on the keyboard changes the ball's location.

1. Develop a data representation for the current position of the ball. The position is best described with a pair of positive integers.

2. Develop a data representation for velocity of the ball. You may assume that the ball always moves exactly 10 pixels at a time but remember that velocity also includes the direction of the movement.
3. Develop a data representation for the ball.
4. The function `ball-image` is written for you. It consumes (the representation of) a ball and produces a rectangle of 300 x 300 pixels with a red dot (diameter 10 pixels) placed at the ball's position.

```
(define a-ball (circle 5 "solid" "red"))

(define (ball-image b)
  (place-image a-ball
               (posn-x (ball-position b))
               (posn-y (ball-position b))
               (empty-scene 300 300)))
```

5. Design the function `ball-next`, which consumes (the representation of) a ball and create a ball that represents where it will be after one "tick."
6. Design a function `ball-change`, which consumes a ball and a "key event," which represents the user hitting a key on the keyboard. If the key event represents the user hitting the up-arrow key, then `ball-change` produces a new ball which is just like the input ball, except that the new ball is moving upward; similarly for key events representing the left-, right- and down-arrow keys: they cause the resulting ball to be one that is moving left, right, or down, respectively. Passing any other keystroke to `ball-change` causes the ball to be unchanged. You can read about key events in the help desk—the `on-key` clause of `big-bang` is a good place to start.
7. Optional: After you have developed these functions, you may wish to add the following lines to your program:

```
 (big-bang    some-initial-ball
              (on-tick ball-next 1/28)
              (to-draw ball-image)
              (on-key ball-change))
```

If you do, be sure to delete them before you turn in your solution. We will subtract points if you leave them in.