CS 2500, Spring 2014
Problem Set 4

Due date: 7pm Tuesday February 4, 2014

Programming Language: BSL

Purpose: to practice designing data representations using unions of classes, and functions for processing unions.

Finger Exercises HtDP/2e: 87, 88, 89, 90, 115, 116, 117, 118, 121, 123, 124, 125, 127, 128

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:

A `Book` is one of:

- `Hardcover`, which has the properties: title, author, number of pages, genre, and a Boolean determining whether this particular book is on the bestseller list;
- `Ebook`, which has the properties: title, author, size in kilobytes, and a symbol designating the ebook reader as 'Kindle, 'Nook, or 'iPad;
- `Audiobook`, which has the properties: title, author, length in minutes, and the name of the person whose voice is recorded.

a) Develop data and structure definitions for a `Book`
b) Design the function `length-of-book` that, given a Book, produces length of the book. For a hardcover book it should produce the number of pages. For an audiobook it should produce the length in minutes. For an ebook, it should produce the size in kilobytes.
c) Design the function `same-author?` which determines if two given books have the same author.

Problem 2:

A movie store sells two kinds of movies: regular and classic. For regular movies, they track the product id (a string, such as "234-87-1DX"), its base price, and the number of years it has been in their collection. For each year a movie is in stock, it is marked down by 3.5% of the base price, but no movie is sold for less than $2. For classic movies, they track the product id (again, represented as a string), and its price, which is never discounted.

a) Design data and structure definitions for the store's items

b) Design `movie-price`, a function that computes the current price of an item. (Think of the program in a cash register and how it computes the current price from the price tag.)

Problem 3:

Below is a data definition for a class of shapes (See the code at the bottom).

a) Add interpretations for the Square and Rectangle classes.

b) Design `shape-shift-x`. The function consumes a Shape, *sh*, and a number, *delta*. It produces a shape that is like *sh* but shifted by *delta* pixels along the x-axis.

c) Design `shape-in?` which consumes a Shape, *sh*, and a Posn, *p*, and determines whether *p* is inside (or on the boundary) of *sh*.

Domain Knowledge: for a point to be within a circle, its distance to the center must be smaller than (or equal to) the radius. For a point to be within a rectangle, its x coordinate must be between the x coordinate of the left line and the x coordinate of the right line. How do you compute the x coordinates of these lines? Naturally something analogous must hold for the y coordinates. Remember that squares are just special rectangles.

d) Design `shape-grow`. The function consumes a Shape, *sh*, and a positive number and produces a new shape that has increased in size by the given increment.

e) Design `shape-draw`. The function consumes a Shape, *sh* and a Scene, *sc* and adds *sh* to *sc*.

```
;; Shape is one of:
;; -- Circle
;; -- Square
;; -- Rectangle

(define-struct circl (x y r outline c))
;; A Circle is a
;;  (make-circl Number Number Number Boolean Symbol)
;; interpretation: x and y determine the center of the circle,
;;   r the radius, outline whether it's outlined or solid,
;;   and c its color

(define-struct squar (x y size outline c))
;; A Square is a
;;  (make-squar Number Number Number Boolean Symbol)
;; interpretation: Supply a good interpretation of Square.

(define-struct rect (x y width height outline c))
;; A Rectangle is a
;;  (make-rect Number Number Number Number Boolean Symbol)
;; interpretation: Supply a good interpretation of Rectangle.
```

Problem 4:

Add a happiness gauge similar to the one from Problem Set 2 (Problem 4) to the chameleon animation from Problem Set 3. With each clock tick, happiness decreases by -0.1, starting with 100, the maximum score; it never falls below 0, the minimum happiness score. To make the chameleon happy, you feed it by pressing the down arrow and two points are added to its happiness. If happiness reaches zero, the chameleon dies and the world should end.
Hint: You will need to define several helper functions.