CS 2500, Spring 2014
Problem Set 3

Due date: 7pm Tuesday January 28, 2014

Programming Language BSL

Purpose The purpose of this problem set is to internalize design skills. Topically, the problem set focuses on functions that process structures and arbitrary unions of data.

Finger Exercises HtDP/2e: 65, 68, 69, 86, 94, 102, 106, 107, 109, 113

You MUST follow the Design Recipe when you develop functions in this problem set. Name your functions using the names given below.

Problem 1:

Consider the following structure definitions:

```
(define-struct painting (artist year style))
(define-struct album (title singer genre))
(define-struct senator (state party))
(define-struct person (name mother father height))
(define-struct shoe (style brand color))
```

a) What are the names of the constructors, predicates and selectors that each of the structures adds to Racket?
b) Provide data definitions for the structure definitions above. Make appropriate assumptions about what data goes with each field.
c) Develop templates for functions that consume the structures above.

Problem 2:

a. Provide a structure type definition and a data definition for representing 4-digit PIN numbers. A PIN number consists of digits represented with a number from 0 to 9.
b. Design the function `difference`. It consumes two representations of PIN numbers and creates a new PIN number from the differences. For each position in the PIN, it subtracts the lesser number from the greater. If the two are the same the new digit is 0. Note: The problem statement mentions two different tasks: one concerning PINs and one concerning digits. This suggests that you design two functions.

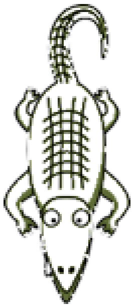
Problem 3:

a. Provide a structure type definition and a data definition for representing points in time since midnight. A point in time consists of three numbers: hours, minutes and seconds.
b. Design a function `time->seconds` which consumes instances of a point in time and produces the number of seconds that have passed since midnight.
c. Design a function `advance-time` which consumes instances of a point in time and produces the point in time one second later.
d. Design a function `time->text` which consumes instances of a point in time and renders the time to look like the display of a common alarm clock. Hint: a text is an

image, *not* a string, but you will need a string version of the time, too. See HelpDesk for more on the text function.

Problem 4:

The drawing of a chameleon in Section 5.7 of HtDP/2e is a **transparent** image. To insert it into DrRacket, copy and save the image from HtDP/2e and insert it with the “Insert Image” menu item. Using this instruction preserves the transparency of the drawing’s pixels.

(define cham )

When a partly transparent image is combined with a colored shape, say a rectangle, the image takes on the underlying color. In the chameleon drawing, it is actually the inside of the animal that is transparent; the area outside is solid white. Try out this expression in your DrRacket, using (`require 2hdtp/image)` :

```
(overlay
  cham
  (rectangle (image-width cham)
             (image-height cham) "solid" "red"))
```

Design a world program that has the chameleon continuously walking back and forth across the screen (you should rotate the image 90 degrees so it is facing right). It starts walking from left to right and when it reaches the right end of the screen, it flips horizontally and begins walking from right to left and switches direction again when it reaches the left boundary.

Of course, like all chameleon’s, ours can change color, too: the key `"r"` turns it red, `"b"` blue, and `"g"` green.

Start with a data definition, `VCham`, for representing chameleons. You will need to represent the position, color and direction of the chameleon in your definition.

Here is a wish list for the functions you will need:

```
draw-cham – renders the world as a chameleon facing left or
right
next-cham – compute the new location of the chameleon in one
clock tick from now

change-cham – changes the color of the chameleon if the “r”, “g”
or “b” key is pressed
```