CS 2500, Lab 5: Lists! Lists! Lists!

This lab is all about Lists and structures. We'll be using them for the rest of the semester (and you saw them on the exam), so it is important that you practice, practice, practice! Be sure to switch roles often. This lab has been designed so you can switch roles after each exercize (or every two if you need more practice).

Part I: Finger Exercises

We start with finger exercises today; lists are important, and they should be second nature.

Consider the following data definition:

```
;; An LoN is one of:
;; - empty
;; - (cons Number LoN)
```

Exercise 1: For each of the following write the corresponding data definition (like above), then make 3 examples of each.

- List of Strings
- List of Images (rectangles, circles, etc)
- List of List of Numbers (LoLoN). Hint: This is a List where the first of a cons is actually an LoN.

Part II: Calisthenics

Exercise 2: Write the function `sum` that consumes a list of numbers and returns the sum of its elements.

Exercise 3: Write the function `product` that consumes a list of numbers and returns the product of its elements.

Exercise 4: Design the function `join-los` that consumes a list of strings and concatenates them. For example:

```
(check-expect
   (join-los (cons "An" (cons "Lo" (cons "S" empty)))) "AnLoS")
```

Part III: Fun with Gravity (yay!)

Here a data definition to get us started. Mak sure you understand the definition/interpretation before moving on.

```
;; A Ball is:
;; - (make-ball Number Number Number Number)
(define-struct ball (x y vx vy))

;; Interpretation:  A Ball represents an object at position X/Y with
;; a velocity VX in the X direction, and VY in the Y direction.

;; All numbers are Pixel units, and computer graphics coordinates

;; Constants
(define WIDTH 400)
(define HEIGHT 400)
(define GRAV-ACC 3)
(define SIZE 10)
```

Exercise 5: Write a data definition for Lists of Balls (LoB).

Exercise 6: Write two templates: one for functions that consume a `Ball`, and another for functions that consume an `LoB`. *Hint:* use the `Ball` template in the `LoB template`.

Coding Warm-up...

Exercise 7: Design the function `off-screen?` that determines whether or not the given `Ball` is off the screen (use `WIDTH` and `HEIGHT` so we can change it later).

Questions: When is an x/y coordinate off screen? Which template should you use?

Exercise 8: Design the function `gravity` that consumes a `Ball` and creates a new one whose Y velocity is *increased* by GRAV-ACC (other components remain the same).

Exercise 9: Design the function `move` that consumes a `Ball` and moves it's X/Y coordinates by the Ball's corresponding velocities (i.e., updates the X/Y by adding VX/VY).

Now for Lists...

Exercise 10: Design the function `draw-lob` that consumes an `LoB` and returns a `Scene` (remember `WIDTH` and `HEIGHT`?) that has a circle of size `SIZE` for each ball in the list, at the right

coordinates.

Exercise 11: Design the function `on-screen` that consumes an `LoB` and filters out all the `off-screen?` Balls.

Exercise 12: Design the function `gravity-all` that consumes an `LoB` and applies `gravity` to all its elements, returning the resulting `LoB`.

Exercise 13: Design the function `move-all` that consumes an `LoB` and applies `move` to all its elements, returning the resulting `LoB`.

Run program...Run!

Here's some code to finish off the exercises... Run it... what does it do?

```
;; mouse :  LoB Number Number MouseEvent -> LOP
;; Add a new random ball
(define (mouse lob x y me)
   (cond [(string=?  me "drag")
      (cons (make-ball x y
                  (- (random 9) 4)
                  (- (+ (random 10) 10)))
            lob)]
          [else lob]))

;; tick :  LoB -> LoB
;; Move, gravitize, then filter out all
;; the off-screen Balls
(define (tick lob)
   (on-screen (move-all (gravity-all lob))))

(define last (big-bang empty
                 (on-mouse mouse)
                 (to-draw draw-lob)
                 (on-tick tick)))
```

If you finish early...

Try adding a color to the Ball structure, and change your functions to match (make sure you draw each Ball with the correct color!). Then change `mouse` to create the new Ball with a random color.

Hint: make a function that returns a color `String` given a number (say it returns one of 5 preselected colors). Then call that function with `(random 5)` to get a random color when needed.