CS 2500, Lab 3

Pair Programming

The lab assignment is to be completed in pairs. Each pair works at one computer. Remember, in pair programming, one member of the team is the pilot and the other is the co-pilot. The pilot does the typing but the co-pilot drives the process. Even though the pilot is the only one typing, both partners should be active in trying to come up with solutions to the exercises. **Make sure to switch roles when indicated in the lab!**

Don't delete your work!
In previous labs some students deleted their solutions for exercises after completing themdon't do this! It is common for exercises to make use of functions or templates defined in earlier exercises.

Part I: Stucture Definitions

For each of the following word problems, extract *structure definitions* for the data involved. You don't need to write tests for them, but be sure to test them informally by trying them out.

Remember structure definitions from class?
`(define-struct name (field ...))`

Exercise 1: The Boston Zoo keeps track of information for every animal that is kept there. For each animal, they store its name, species, age, breakfast hour, and dinner hour (if they don't get fed twice a day, they try to eat the visitors...).

Exercise 2: Each zoo attendant has a name, and is assigned to watch over exactly three different animals.

Part II: Data Definitions

Take your structure definitions from above and add data definitions. (E.g. ;; An animal is...) Recall that a data definition for structured data states, in a mixture of English and Racket, how we construct elements of this class of data, including what kind of data each structure field contains. Refer to Section 6.4 of HtDP if you need to review!

Part III: Templates

Construct a template for each of the data definitions from Part II. Recall that the template describes what we know about the input to a function (what accessors we can use on them). If you need to review templates for compound data, look in Section 6.5 of HtDP.

Part IV: From Templates to Functions

Switch Roles! Consider the templates you built from the data definitions and design the following functions based on them. **You must formulate examples and tests!**

Exercise 3: Using your data definition, structure definition and template from above, develop a function that takes an animal, and returns the animal with 1 added to its age (presumably the function will be used on its birthday).

Exercise 4: Using your data definition, structure definition and template from above, develop a function that takes an animal and the current hour, and returns whether it's mealtime.

Exercise 5: In preparation for next April Fool's Day, the system manager of the zoo wants you to design a function that takes an animal, and returns the animal with its age converted to *dog years*. Note there are 7 dog years in **1** human year.

Exercise 6: Using your data definition, structure definition and template from above, develop a function that takes an attendant, and returns the total age of the animals that the attendant has been assigned to.

Part V: Challenge Problem

Switch Roles! Some of the TAs (oddly, they've asked to remain anonymous) are working on a game that they call *Chip, the Cheap Sheep*. So far, they've put together a few frames of animation for it:

0 1 2 3

Your goal is to create a simple proof-of-concept game engine: Chip will run from offscreen to the point the user clicks on.

Use the design recipe when designing functions!

Exercise 7: Create a function named `which-chip` that takes a `number`, and returns the corresponding image from the sequence above.

This can be done several ways... which one is best?

Clicking on any of the images above will pull up a webpage with the frames. You should be able to drag the images from the webpage into DrRacket, once they are there you should know what to do. If not save them to the desktop, and use "*Insert*" > "*Insert Image...*" from the DrRacket menu bar.

Exercise 8: Write a data and structure definition for your `world`. You'll want to be able to know Chip's coordinates, the coordinates he is running to, and which `frame` of the animation he is currently on.

While you're at it, write a template for functions that take a world, and define a variable named `world0` that is your initial world (start by assuming the user clicked in the center, and Chip is just offscreen).

Exercise 9: Write a function named `draw-chip` that takes a `world` and places the image of Chip's current frame at his current coordinates into an `empty-scene` of size 400x400. (Use your `which-chip` function!)

Exercise 10: Write a function named `move-chip` that takes a `world` and returns a `world`; moving Chip to the *left* by some amount. (It looks like he's going pretty fast!)

This assumes he starts at the y-coordinate of his destination, and only has to go left.

If he gets to his destination, have him stop moving.

Exercise 11: **Switch Roles!** Write a function named `next-chip` that takes a `world` and returns a `world`; incrementing the `frame` field and *wrapping* the number so it cannot be greater than 3.

The `remainder` function can be used for this:

```
(remainder 1 4) ;; ==> 1
(remainder 2 4) ;; ==> 2
(remainder 5 4) ;; ==> 1
(remainder 6 4) ;; ==> 2
```

Exercise 12: Now, make Chip respond to mouse clicks. Write a `clack` function that takes a `world`, x and y coordinates, and a `mouse event`, and returns a new `world`, with the mouse

coordinates as Chip's new destination, and with Chip teleported offscreen (presumably at the same y-coordinate as his destination), so that he can run to the point the user clicked on.

Remember that mouse events are just strings. Be sure to ignore all mouse events except `"button-down"`.

Exercise 13: Put all this together with a *big-bang* and see Chip the Cheap Sheep run!

```
(define (tock world)
   (move-chip (next-chip world)))


(big-bang world0
   (on-draw draw-chip)
   (on-tick tock)
   (on-mouse clack))
```

Exercise 14: If you still have extra time, try variations: maybe make a bouncing ball for Chip to chase.

That's All Folks

If you had trouble finishing any of the exercises in the lab or homework, or just feel like you're struggling with any of the material, please feel free to come to office hours and talk to a TA or tutor for additional assistance.