Comp160
Lab 6
Spring 2018

In this lab you'll be working with itemizations. These are covered in Section 4 of the text. Of particular importance is Section 4.6 as it covers the application of the design recipe to itemized data. You'll also be called on to work with World Programs. If you need or want a refresher on the structure of a World Program, then refer back to Section 3.6.

Lab 6

From here on out your main goal is to practice the systematic design and development of functions and programs through the application of the design recipe and iterative refinement. If you're arriving at finished definitions by some means other than the recipe, then you're doing it wrong. Your code should not only do what it's supposed to do but should, in its logic, exhibit a reasoned and explainable structure that is consistent with known best practices in programming. You can achieve this through the design recipe. Practice and perfect the application of the design recipe now even if you think you don't need it. You can choose to use it or not use it in the future. You'll also be tested on the recipe, so there's that.

1. (Enumerations) Start with Exercise 50 and Exercise 51.

   For these exercises you already have a data definition for an enumeration, *TrafficLight*, and a fully designed and implemented function on that enumeration, *traffic-light-next*, ready to go. Exercise 50 asks that you finish testing *traffic-light-next* and exercise 51 has you put this function to use in a world-style animation program. You already have the "WorldState" definition, TrafficLight, and the clock-tick event handler function, traffic-light-next. You should only need design a draw-event handler function and then create a *main* to through which you'll kick start big-bang[1].

   Diligently follow the design recipe to create your draw-event function. With enumerations we have something to do at each step of the recipe; don't skip anything. If you're not sure how to proceed at any one step of the recipe or if you'd like me to check your work, *ask for help*. Be certain to hit Run after each step of the process to check that you haven't introduced any syntax errors.

[1] Review 3.6 or the squish program if you need a refresher on setting up the main function

2. (Intervals Warm-up) Before moving into intervals, review and check your understanding of interval notation by doing Exercise

52. Write you answer as a comment in your program. Label it and place it below your traffic light program.

3. (Intervals) The first edition of HTDP comes along with a set of practice problems. You can find them here.[2] We'll use this one from section 4 to practice intervals.

   > A manufacturing company measured the productivity of its workers and found that between the hours of 6am and 10am they could produce 30 pieces/hour/worker; between 10am and 2pm they could produce 40 pieces/hour/worker; and between 2pm and 6pm they could produce 35 pieces/hour/worker.
   >
   > Develop a function that takes an hour of the day between 6am and 6pm, in twenty-four hour format, along with the number of workers and computes the total number of pieces produced during that hour.

   Work on diligently following the design recipe. You'll need to start this problem from step 1 by writing a data definition. The information at the core of this problem is the current *Time*. It needs defining as data in our program. It is clearly an interval as it's numerical data broken down into ranges of values. Write a data definition for *Time* and proceed on from there. Once again, run your program after every step of the recipe to check for errors and if you're not sure what to do with a step or want some verification that you're proceeding on track, ask for help.

[2] These problems come with solutions but the solutions don't show you how you can arrive at them through adherence to the design recipe which is ultimately more useful to you.

At this point you're done with lab 6. Print your work and turn it in. Hopefully you have some time left in lab. If you do, then I recommend you use that time to practice function design and check with me to see if you're properly working the design recipe process. Check below for some recommended activities.

Study for the next Exam

On Wednesday you have an exam over the design of basic functions. That means no itemizations which means information that can be easily represented with built in data types: strings, numbers, booleans, and images. This also means no conditionals necessary[3]. An excellent way to study is to walk through the design recipe process using functions like those seen in Section 3.2. Don't be afraid to redesign these functions if you've seen them before. You should be able to explain every bit of the final product in terms of the design recipe.

[3] you might,however, choose to use them to handle boolean valued functions even though its not strictly needed

If you're looking for something more or something different, then here are a few more practice problems. The first few will work your knowledge of boolean valued functions, a.k.a. *predicate functions*, and operations. I recommend you attempt to complete them without resorting to conditionals or if expressions. Instead, you should use boolean operations like *and*, *or*, and *not*. Even better, do them both ways.

- Design a predicate function that checks to see if a number is a multiple of 5. (Hint)
- Design a predicate function that checks if a number is a multiple of 3 or a multiple of 7, but not a multiple of both.
- Design a function that takes two images and determines if the first contains at least half as many pixels as the second.
- Design a function that draws a bullseye image composed of three circles a la the Target logo. Users should be able to specify the color of each circle.

Looking Ahead

If you're feeling good about the design recipe and would rather work more with itemizations, then start working your way through Section 4.7. This section introduces you to Finite State Machines, a fundamental idea in computing and one that is closely tied to itemization based design.