Comp160
Lab 8

Spring 2018

In this lab you'll cut your teeth on reasoning and programming with *struct* type data. The exercises below as well as all the details about structs that you need can be found in Chapter 5.

Lab 8

We'll cover sections one and two of chapter 5 in class. Review them as needed before moving on with the lab.

Programming with posn

Give section 5.3 a quick read. Then proceed with the following:

1. Do Execise 63. First run the computation and find out the result[1]. Then formulate each step of the computation as a unit test. For example, if we wanted to trace $(+(-42)4(*52))$, which results in 16, then we'd write:

```
(check-expect (+ (- 4 2) 4 (* 5 2)) 16)
(check-expect (+ 2 4 (* 5 2)) 16)
(check-expect (+ 2 4 10) 16)
(check-expect 16 16)
```

   Each of these tests should pass. After all, how the computation unfolds has no bearing on the final result. What these tests don't tell you is if you got the number of steps and order of steps correct. To verify that you need to use the stepper. When running the stepper comment out all but the first step, the one with the original expression, and then step through that test. After checking your work with the stepper, comment about where you got off track and offer some explanation for how and why the actual computation differed from your expectation.

2. Do Exercise 64. Include the signature, purpose statements, and tests with your answer. Write up the test case given in the problem as a unit test and provide two other test cases. Be sure to answer the question about the strategy used in a comment under your function definition.

[1] Better yet, do the computation by hand using pencil and paper.

Defining Structures

Section 5.4 tells you everything you need to know about defining your own struct types. Notice that this is a new, third type of define expression[2]. Read along with the section and do the following:

[2] Constants, functions, and now structs

3. Do Exercise 65 and Exercise 66 but only for the *pet* and *cd* structure types. Rather than simply write down the function names and the types of the fields as requested by the exercises, let's write up our answers in a way the computer can check. First copy the aforementioned structure definitions from exercise 65 into the DrRacket Definition's window. Next, answer exercise 66 by documenting the field types as a comment above the definition[3] and then write out not 1 but 2 concrete examples below the definition. Out of necessity, these examples demonstrate the constructor as this is the only way we have to specify structure data. Finally, write unit tests that demonstrate all the remaining functions discussed in exercise 65. For *posn* structures this would look like:

[3] this is a proto-data definition for the structure type

```
; a posn has two fields, x and y. They should both be numbers.
(define-struct posn [x y])
;; constructed examples
(make-posn 3 2) ; the position (3,2)
(make-posn 0 1) ; the position (0,1)
;; selectors
(check-expect (posn-x (make-posn 3 2)) 3)
(check-expect (posn-y (make-posn 0 1)) 1)
;; predicate
(check-expect (posn? (make-posn 0 1) #true)
(check-expect (posn? 4) #false)
```

4. Now skip ahead to the start of section 5.5. Here we see the visual representation of structures as boxes. *After you've completed lab and printed your definitions,* carry out Exercise 69 for pet and cd structure types you dealt with in the previous problem by sketching the box representations of your concrete examples next to the constructed examples on your printed definitions.

Bouncing Balls

As we're already aware, there are often multiple ways to represent the information of our problem with data in our program. This continues to be true with structure data. Our world programs will make good use of objects moving freely around a 2D space much like the bouncing balls discussed after exercise 66 in section 5.4. Return there and start reading. Then do the following:

5. Carry out Exercise 67. Start by copying the given code to your definitions window. Then construct two more examples of *balld* structures. Above each example write a comment that describes how to interpret the example in terms of bouncing balls. Finally, write a comment that discusses the limitations of this representation choice. Are there types of balls it cannot represent?

6. Carry out Exercise 68 by once again copying the structure definition into your definition and then creating three concrete examples: the *ballf* equivalent of the previously defined ball1 and two more examples of your own choosing. Above each example provide a comment that describes how to interpret the structure as a ball. Finally, compare an contrast the three ball structures defined in this section:*ballf*, *balld*, and *ball*.

7. After you've printed your definitions, draw box representations of ball1 and it's *ballf* type equivalent from the previous problem.

You're done. Remember that several lab problems asked you to draw something on your work after you printed it, so print and then go back to complete those problems as directed. If you have time left in lab, then I suggest you get started on the homework.