CSE 341 — Racket Discussion Questions Part 2

These questions deal with structs, representing objects, lexical scoping, and macros.

1. Define a `struct` called `point3d` that represents 3D points. Create a point `p` at the origin; change its z value to be 10; and print it out. It shoud print as `(point3d 0 0 10)`.

2. Define a `make-cell` function that returns a simulated instance of a cell with a single field `value`, which should be hidden (using lexical scoping). The cell should provide "methods" for `get-value` and `set-value!`. Follow the bank account example in doing this. The value should start out as null.

3. Similarly but with more bells and whistles . . . define a `make-point` function that returns a simulated instance of point with x and y fields, which should be hidden (using lexical scoping). The point should provide "methods" for `get-x`, `get-y`, `set-x!`, `set-y!`, and `print-point`. Follow the bank account example in doing this. The fields should start out as 0.

4. What does this expression evaluate to? Why? (What environment is `(f 3)` evaluated in? What environment is the body of the lambda evaluated in?)

```
(let ((x 2))
  (let ((f (lambda (n) (+ x n))))
    (let ((x 17))
      (f 3))))
```

5. What does this expression evaluate to? Why?

```
(define (addN n)
  (lambda (m) (+ m n)))

(let* ((m 10)
       (n 20)
       (addit (addN 3)))
  (addit 100))
```

6. Define a Racket macro `and2` that is a 2-argument version of `and`. Hint: the value of the `and` expression in Racket is the value of the *last* subexpression if all of them are something other than `#f`. The `and2` macro should work the same, so `(and2 #t "squid")` should evaluate to `"squid"`.