1. What do the following Racket expressions evaluate to?
   (a) (* 2 (+ 4 5)) => 18
   (b) (= 3 (+ 1 3)) => #f
   (c) (car '(elmer fudd daffy duck)) => elmer
   (d) (cdr '(elmer fudd daffy duck)) => (fudd daffy duck)
   (e) (and (= 1 2) (= 10 (/ 1 0))) => #f

2. Find the squid! For each of the following variables, write an expression that picks out the symbol squid. For example, for this definition: (define x '(squid clam octopus)) the answer is (car x).
   (a) (define y '(clam squid octopus)) => (cadr y)
   (b) (define z '(clam starfish (squid octopus) mollusc)) => (caaddr z)

3. Define a Racket function to find the average of two numbers.
   (define (average x y)
     (/ (+ x y) 2.0))
   You can also use the integer 2 as the constant — in that case you’ll get a fractional result rather than a float if you find the average of an even and an odd number. (Try it.)

4. Define a Racket function mymax to find the maximum of two numbers.
   (define (mymax x y)
     (if (> x y) x y))

5. Suppose we evaluate the following Racket expressions:
   (define x '(snail clam))
   (define y '(octopus squid scallop))

   Draw box-and-arrow diagrams of the result of evaluating the following expressions. What parts of the list are created fresh, and which are shared with the variables x and y?
   (a) (cons 'geoduck x))
   (b) (cons y y)
   (c) (append x y)
   (d) (cdr y)

6. Define a recursive function sum to find the sum of the numbers in a list.
   (define (sum s)
     (if (null? s)
         0
         (+ (car s) (sum (cdr s)))))

7. Define a tail recursive version of sum. (Define an auxiliary function if needed.)
   (define (sum s)
     (sum-helper s 0))
   (define (sum-helper ssofar)
     (if (null? s)
        sofar
         (sum-helper (cdr s) (+ (car s) sofar))))

8. What is the result of evaluating the following Racket expressions?
(a) (let ((x (+ 2 4))
   (y 100))
   (+ x y))

106

(b) (let ((x 100)
   (y 5))
   (let ((x 1))
     (+ x y)))

6

9. Define a function mylength to find the length of a list.

(define (mylength s)
  (if (null? s)
      0
      (+ 1 (mylength (cdr s)))))