

EECS 111 Quiz #1

February 3, 2010

***Don't panic!** Read each question through. If any part confuses you, come up and ask me privately. Watch your time. Don't spend forever on any one question. Write cleanly. If you need to make big changes, X out the current code, write "see back" and write your new version on the back, with the number of the question.*

1. (10 pts) The following code was supposed to return a person's BMI (body mass index) category, using the formula $BMI = \text{weight} * 703 / \text{height}^2$, where weight is in pounds and height is in inches, and the table on the right. But the code has many errors. Circle each error, and say briefly what's wrong. On the right, write the correct code.

$bmi < 18.5$	underweight
$18.5 \leq bmi < 25$	normal
$25 \leq bmi < 30$	overweight
$30 \leq bmi$	obese

```
(define (bmi wgt hgt)
  (bmi-cat
   703 * wgt / hgt * hgt))

(define (bmi-cat x)
  (cond (x < 30) "overweight"
        (x < 25) "normal"
        (x < 18.5) "underweight"
        otherwise "obese"))
```

2. (10 pts) What does `fn1` in the code below return for the following input values?

<code>(fn1 (list 1 2 3 4))</code>	
<code>(fn1 (list 4 3 2 1))</code>	
<code>(fn1 (list (list 1 2) (list 3 4)))</code>	
<code>(fn1 (list 1))</code>	
<code>(fn1 empty)</code>	

Describe what `fn1` returns in general:

```
(define (fn1 l)
  (fn2 l empty))

(define (fn2 l1 l2)
  (if (empty? l1) l2 (fn2 (cdr l1) (cons (car l1) l2))))
```

3. (10 pts) What does `fn1` in the code below return for the following input values?

<code>(fn1 (list 1 2 3))</code>	
<code>(fn1 (list 1))</code>	
<code>(fn1 (list 4 3 2 1))</code>	
<code>(fn1 empty)</code>	

Describe what `fn1` returns in general:

```
(define (fn1 l)
  (fn2 l (/ (fn3 l) (length l))))

(define (fn2 l x)
  (if (empty? l) empty (cons (- (car l) x) (fn2 (cdr l) x))))

(define (fn3 l)
  (if (empty? l) 0 (+ (car l) (fn3 (cdr l)))))
```

4. (10 pts) Define a function `(next-pow m n)` for non-negative integers m and n to return the smallest k such that $m^k \geq n$. Feel free to define helper functions. Some test cases:

```
(check-expect (next-pow 2 8) 3)
(check-expect (next-pow 2 9) 4)
(check-expect (next-pow 2 16) 4)
(check-expect (next-pow 2 1) 0)
(check-expect (next-pow 3 1) 0)
(check-expect (next-pow 3 27) 3)
(check-expect (next-pow 3 80) 4)
```

5. (10 pts) Define a function `(merge list1 list2)` to return the sorted merger of two sorted lists of numbers, including duplicates. Some test cases:

```
(check-expect (merge empty empty) empty)
(check-expect (merge (list 1 2 3) empty) (list 1 2 3))
(check-expect (merge empty (list 1 2 3)) (list 1 2 3))
(check-expect (merge (list 1 2 2 8) (list 1 1 5))
              (list 1 1 1 2 2 5 8))
```