Week 4 – 04-02-12
Midterm 1 Review

Topics to Cover:

1. Scheme
2. Higher Order Functions
3. Normal vs. Applicative
4. Recursive vs. Iterative Processes
5. Big O
6. Programming Methodology
7. Project Questions

Scheme Questions

What will Scheme print in response to the following expressions? If an expression produces an error or runs forever without producing a result, just say "error"; If the value of an expression is a procedure, just say "procedure."

```
(word '(+ 2 3) (+ 2 3))

((lambda (x y z) (* 5 y)) 3 4 7)

((if 3 - *) 23 2)

(lambda (x) (/ x 0))

(butfirst '(help))

(let ((+ -))
  (+ 8 2))

(every – (filter number? '(the 1 after 909)))

(let ((a 2) (b (+ a 3)))
  (word a b))

((lambda (a b c) (b c a)) 1 + 4)

(se ('tell 'me 'why))

(every pigl (se '() (word 61 'a) (se 'is 'a) 'great '(class)))
```

**First express it in lambdas…then head into the evaluation

```
(let ((a (square 2))
      (b (+ 3 4)))
  (let ((c (+ a (let ((d 3))
                  (+ b d))))
        (e 14))
    (* (+ a b) (- e c))))
```

Higher Order Function

Write a procedure called **make-manip** which takes two procedures, *pred* and *manip* and returns a manipulator! A manipulator is a procedure that takes a sentence as its argument and returns a sentence in which every element from which *pred* returns true is manipulated with *manip*, and all of the other elements are the same. For example:

```
>((make-manip odd? 1+) '(3 6 9 12))
   (4 6 10 12)
```

No Helper functions!

Write one version using HOF and no explicit recursion.

Write another using no HOF.

Normal vs. Applicative Order
True or False:

(define (f x) (* x x x))

Evaluating (f (g y)) evaluates (g y) more often in applicative order than in normal order.

Suppose you were given the following definitions:
(define (double x) (+ x x))
(define (foo x y z) (+ x y z)
(define (bar x y z k) k)

How many times is + called for
(foo (double (+ 1 1)) (double (+ 1 1)) (+ 1 1))

…under normal order?

…under applicative order?

How many times is + called for
(bar (double (+ 1 1)) (double (+ 1 1)) (+ 1 1) 1)

…under normal order?

…under applicative order?

Big O
True or False:
If foo is Theta(n) and bar is Theta(n^2), then you can always compute (foo 1000) faster than (bar 1000) on the same computer.

Given These Definitions:

```
(define (f x)
  (if (< x 0)
      1
      (f (- x 3))))

(define (g y)
  (if (< y 104)
      0
      (* (f y) (f (- y 4)))))
```

```
(define (h z)
   (if (< z 4)
       0
       (+ (h (- z 2))
          (h (- z 1)))))
```

State whether or not these statements are true or false:

h generates an iterative process (i.e. uses Θ(1) space)

f is Θ(x).

h is Θ(z2)

f and g have the same order of growth

g and h have the same order of growth

Project Questions
Write a strategy four-cards that hits only if a player has fewer than four cards

Write a procedure n-cards that takes an argument n and returns a strategy that hits only if a player has fewer than n cards

Recursive vs. Iterative

Write a procedure **(insert value insert-before sent)** that'll return a sentence with 'value' inserted in the list (counting from 1):

(insert 'a 3 '(1 2 3 4)) → (1 2 a 3 4)
(insert x 'a '(a b c d)) → (x a b c d)
(insert a '4 '(1 2 3 4)) → (1 2 3 a 4)

You can assume that **insert-before** will always be in the list. You may not use any mutators (if you know of them)

Write a version using a recursive process....

and another with an iterative process.

Programming Methodology
Greg wanted to write a procedure that would split a non-empty word into a sentence of consecutive, identical letters as follows:

(split 'aaabbcdddaa) → (aaa bb c ddd aa)
(split 'abababab) → (a b a b a b a b)
(split 'aaa) → (aaa)
(split 'a) → (a)

Here's what he wrote:

```
1:  (define (split wd)
2:      (split-help (first wd) (bf wd)))
3:
4:  (define (split-help cur wd)
5:      (cond ((empty? wd) (se))
6:            ((equal? cur (first wd))
7:             (split-help (word cur (first wd)) (bf wd)))
8:            (else
9:              (se cur (split-help (first wd) (bf wd))))))
```

There are two bugs.

Part A:
What does (split 'abc) return?
On which line number is the bug that causes this error? Line ___
What should the line say?

Part B:
Where's the other bug? Line ___
What should the line say?