Recursive vs Iterative Processes
Decide whether each of the following procedures generates a recursive or iterative process.

```
(define (foo x)
  (* (- (+ (/ x 3) 4) 6) 2))

(define (foo x)
  (if (= x 0) 0 (+ x (foo (- x 1)))))

(define (helper1 x)
  (if (= x 0) 1 (helper1 (- x 1))))

(define (helper2 x)
  (if (= x 0) 1 (+ 1 (helper2 (- x 1)))))

(define (bar x)
  (if (even? x) (helper1 (- x 1)) (helper1 (- x 2))))

(define (bar x)
  (if (even? x) (helper2 (- x 1)) (helper2 (- x 2))))

(define (bar x)
  (if (= x 0) (helper2 x) (helper1 x)))

(define (bar x)
  (if (= x 0) (helper1 x) (helper2 x)))

(define (bar x)
  (cond ((= x 0) 1)
        ((= (helper2 x) 3) 5)
        (else (helper1 x))))

(define (bar x)
  (helper2 (helper1 x)))
```

More Recursion Practice
1. There used to be a number 1. Now there isn’t, but I didn’t want to renumber all the questions, so here’s a placeholder.

2. There is something called a falling factorial. (falling n k) means that k consecutive numbers should be multiplied together, starting from n and working downward. For example, (falling 7 3) means 7 * 6 * 5. Write the procedure falling that generates an iterative process.
3. Write a version of (expt base power) that works with negative powers as well.

4. Implement (ab+c a b c) that takes in values a, b, c and returns (a*b) + c. However, you cannot use *. Make it a recursive process.

5. Implement (ab+c a b c) as an iterative process. Don't define helper procedures.

**Order of Growth**

Decide what the order of growth is for each of the following:

```scheme
(define (fact x)
  (if (= x 0)
      1
      (* x (fact (- x 1))))
)

(define (fact-iter x answer)
  (if (= x 0)
      answer
      (fact-iter (- x 1) (* answer x)))
)

(define (sum-of-facts x n)
  (if (= n 0)
      0
      (+ (fact x) (sum-of-facts x (- n 1))))
)

(define (fib n)
  (if (<= n 1)
      1
      (+ (fib (- n 1)) (fib (- n 2))))
)

(define (square n)
  (cond ((= n 0) 0)
        ((even? n) (* (square (quotient n 2)) 4))
        (else (+ (square (- n 1)) (- (+ n n) 1)))))
```