6.090, Building Programming Experience
Ben Vandiver
Lecture 7: Quiz Day

Hypothetical "Implementation" of pairs

```
(define cons
  (lambda (a b)
    (lambda (x)
      (if x a b))))

(define (car p)
  (p #t))
```

Do type analysis on the above. p is located inside a single open paren and is followed by a Boolean. It must therefore be a procedure which takes in a Boolean and returns something (we don't know what).

```
(define (cdr p)
  (p #f))
```

(car (cons 1 2)) => 1
(cdr (cons 1 2)) => 2

((lambda (p) (p #t)) ((lambda (a b) (lambda (x) (if x a b)))) 1 (2))

((lambda (p) (p #t)) (lambda (x) (if x 1 2)))

((lambda (x) (if x 1 2)) #t)

(if #t 1 2) => 1

Three arguments of cons

```
(define cons3
  (lambda (a b c)
    (lambda (x)
      (if (= x 0) a (if (= x 4 2) b c)))))

(define (car p)
  (p 0))

(define (cdr p)
  (p 42))

(define (cgr p)
  (p 7))
```

The three above definitions are nonsensical, created to make a point that we can define functions any way we wish, with any number or value we choose.

Sorting…in ascending order.

```
(define (min x y)
  (if  (> x y)
       y
       x))

(define (min-list lst)
  (if (null? (cdr lst))
      (car lst)
      (min (car lst) (min-list (cdr lst)))))

(define (without-n lst n)
  (if (null? lst)
      nil
      (if (= (car lst) n)
          (cdr lst)
          (cons (car lst)
                (without-n (cdr lst) n)))))

(define (sort-list lst)
  (if (or (null? lst) (null? (cdr lst)))
      lst
      (let ((least (min-list lst)))
        (cons least
              (sort-list (without-n lst least))))))
```

Create a function insert, to insert an element into a list (unsorted), using only one helper function.

```
(define (insert elem lst)
  (if (null? lst)
      (list elem)
      (if (< elem (car lst))
          (cons elem lst)
          (cons (car lst)
                (insert elem (cdr lst))))))

(define (insert-sort lst)
  (if (or (null? lst) (null? cdr lst))
      lst
      (insert (car lst)
              (insert-sort (cdr lst)))))
```