CSSE 304 Day 05 Summary

1. `positives` and `sorted?`

2. lambda is magnificent! I suggest that you not try to copy the transcript from the slide (you can look at it anytime), but instead use this space to make notes on what is happening there.

3. Properties of a first-class data object:

4. Local variables *via* `let` and `let*`

```
(define L '(4 3 2))
(let ([first (car L)]
      [second (cadr L)])
  (list (+ first second) (- first second)))
```

is equivalent to

```
(define L '(4 3 2))
((lambda (first second)
   (list (+ first second) (- first second)))
 (car L)
 (cadr L))
```

What goes wrong with

```
(define xxx
   (lambda (L)
     (let ([a  (car L)]
           [b  (cdr L)]
           [c  (car b)])
       (list c a))))
(xxx '(1 2 3))
```

?

```
(define xxx
   (lambda (L)
     (let* ([a  (car L)]
            [b  (cdr L)]
            [c  (car b)])
       (list c a))))
```

translates to

```
(define xxx
   (lambda (L)
     (let ([a (car L)])
       (let ([b  (cdr L)])
          (let ([c  (car b)])
              (list c a))))))
```

5. A place for your notes about the "Abe Lincoln" `fact` example (code is in the “Live in class” folder)

6. How is letrec different than `let` and `let*`?

7. Translate the named-`let` code below into `letrec`-based code.

```
(define fact
  (lambda (n)
      (let loop ([x n] [prod 1])
         (if (zero? x)
              prod
              (loop (- x 1) (* prod x))))))
```

8. Given the box-and-pointer diagram on the slide, how would Scheme output this object?

Try to write code that creates this object without using `quote.`