CSSE 304 Day 08

1. Extended BNF grammar for s-lists:
   ```
   <s-list>  ::=  ( {<s-exp>}* )
   <s-exp>   ::=  <symbol> | <s-list>
   ```
2. Standard BNF grammar for s-lists:
   ```
   <s-list>  ::=  ()
             ::=  (<s-exp>  . <s-list>)   ; recall that Scheme prints proper lists without dots
   <s-exp>   ::=  <symbol> | <s-list>
   ```

3. We'll write several procedures that take s-lists as arguments. (Live coding, follow along electronically if you prefer)

`(define (contains? slist sym)` ; does `slist` contain `sym`?

`(define (count-occurrences slist sym)` ; how many occurrences of `sym` in `slist`?

`(define (flatten slist)` ; create a 1-level list from the symbols in `slist` (maintain the order)

`(define (notate-depth slist)` ; replace each symbol with a 2-list: the symbol and its depth within `slist`.

`(define (subst s1 s2 slist)` ; substitute `s2` for each occurrence of `s1` in `slist`. Probably an exercise for later.

`case-lambda` provides a more specific interface for writing variable-arity procedures. Examples are in the slides.
Your notes and questions: