0. **Your thoughts?**  If Scheme was a character, what would it look like?
1. A Deep Problem

`deep-squares`, which takes a deep list of numbers and returns a list with each value squared, is given below.

```
(define (deep-squares lol)
  (cond ((null? lol) '())
        ((list? (car lol))
         (cons (map square (car lol))
               (deep-squares (cdr lol)))))
```

For which of the following inputs will `deep-squares` not work as intended?

- (a) `(deep-squares '())` ○ Works ○ Broken
- (b) `(deep-squares '(1 (2 3) 4))` ○ Works ○ Broken
- (c) `(deep-squares '(1 (2 3) ((4)) 5))` ○ Works ○ Broken

Which line number contains the bug? ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

2. ... That Factors Into Your Learning

Implement the `factors` procedure in Scheme, which takes an integer `n` that is greater than 1 and returns a list of all of the factors of `n` from 1 to `n - 1` in increasing order. You may not need to use all the lines.

*Hint*: The built-in `modulo` procedure returns the remainder when dividing one number by the other.

```
(scm> (modulo 5 3)
  2
(scm> (modulo 14 2)
  0

(define (factors n)
  (define (factors-helper i n)
    (if ________________________________________________________________
        nil
        (______________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
        ))))

  (factors-helper _______ _______)
)

(scm> (factors 6)
(1 2 3)
(scm> (factors 7)
(1)
(scm> (factors 28)
(1 2 4 7 14))
```