Goals of this tutorial
You should be able to...
• understand and write data definitions for lists
• understand and use the template for processing lists to write recursive functions consuming this type of data.
• step through list functions.

Midterm Review: Study Tips
But first, a little midterm review!
• re-do assignments.
• read the assignment post-mortems and feedback given to you on your own assignments.
• do the stepping practice problems on the course website.
• for further practice, try problems from the textbook (be aware of differences between our course and the textbook).
• do all the above by hand.
Midterm Review Problem: Design Recipe
Consider the function, mystery:

```
(define (mystery a b c)
  (cond
    [(< a 10) 'no]
    [(<= a 15) 'yes]
    [(even? c) 'no]
    [(symbol? b) 'yes]
    [else (> c 9)])
```

Write the contract that most closely fits mystery.

Midterm Review Problem: Design Recipe
Now, write a complete set of tests that will sufficiently test this function. Write a short comment for each test that describes what part of the function it is covering.

```
(define (mystery a b c)
  (cond
    [(< a 10) 'no]
    [(<= a 15) 'yes]
    [(even? c) 'no]
    [(symbol? b) 'yes]
    [else (> c 9)])
```

Midterm Review Problem - Stepping
The following definitions have been processed in the Beginning Student language:

```
(define u 64)
(define v (sqrt u))
(define (f u v)
  (cond [(and (posn? (make-posn u v)) v) "hello"]
        [(< 0 u) "bye"]))
```

Step through the following:

```
(f v (or (> v u) false))
```
Review: List data definition

;; A (listof Any) is one of:
;; ⋆ empty
;; ⋆ (cons Any (listof Any))

From the data definition, a list of values of any type is either empty or it consists of a first value followed by a list of values (the rest of the list).

This is a recursive definition. It contains a base case, and a recursive (self-referential) case.

Recursive types should be processed with recursive functions.

Group Problem - symbols-equal?

Recall the data definition and template for lists of Symbols from class:

;; A (listof Sym) is one of:
;; ⋆ empty
;; ⋆ (cons Sym (listof Sym))

;; my-los-fn: (listof Sym) → Any
(define (my-los-fn los)
  (cond
    [(empty? los) . . . ]
    [else (. . (first los) . . .
            (my-los-fn (rest los)) . . .)]))

Group Problem - symbols-equal?

Based on the previous template, write a function symbols-equal? that consumes a list of symbols and produces true if all of the symbols are equal, and false otherwise. Include the contract and examples.

Hint: The template includes only one base case, but sometimes functions need multiple base cases.
Stepping Demonstration - condensed trace

\[ (\text{define } (\text{symbols-equal? } \text{los})) \]

\[ (\text{cond}) \]

\[ ([\text{empty? } \text{los}] \text{ true}) \]

\[ ([\text{empty? } (\text{rest los})] \text{ true}) \]

\[ (= \text{else} (\text{and } (\text{symbol}\equiv? (\text{first los}) (\text{first } (\text{rest los})))) \]

\[ (\text{symbols-equal? } (\text{rest los})))))) \]


Using our definition of \text{symbols-equal?}, we will perform a condensed trace of:

\[ (\text{symbols-equal? } (\text{cons } s (\text{cons } s (\text{cons } h \text{ empty})))) \]

⇒ (and (symbol\equiv? s s)

⇒ (symbols-equal? (rest (cons s (cons s (cons h empty)))))))

⇒ (and true

⇒ (symbols-equal? (rest (cons s (cons s (cons h empty)))))))

⇒ (and (symbols-equal? (cons s (cons s (cons h empty))))))

⇒ (and (and (symbol\equiv? s h)

⇒ (symbols-equal? (rest (cons s (cons h empty)))))))

⇒ (and (and false

⇒ (symbols-equal? (rest (cons s (cons h empty)))))))

⇒ (and false)

⇒ false


Group Problem - list of Posn
Recall the data definition for the built-in structure Posn:

;; A Posn is a (make-posn Num Num)

Write a data definition and template function for a list of Posn structures.
Group Problem - on-axes
Using your template for a list of Posns, write a function on-axes that consumes a list of points (as Posns), lop, and produces a list of those points in lop that lie on the x or y axes. Include examples.