Goals of this tutorial
You should be able to...

- write functions that process two lists at different rates.
- understand limitations of quote lists.
- understand and process two-dimensional data represented by nested lists.
- understand and process association lists.

Testing list equality
;; (list=? lst1 lst2) determines if lst1 and lst2 are equal
;; list=?: (listof Num) (listof Num) → Bool
(define (list=? lst1 lst2)
  (cond
   [(and (empty? lst1) (empty? lst2)) . . .]
   [(and (empty? lst1) (cons? lst2)) (. . . (first lst2) . . . (rest lst2) . . . )]
   [(and (cons? lst1) (empty? lst2)) (. . . (first lst1) . . . (rest lst1) . . . )]
   [(and (cons? lst1) (cons? lst2)) ??? ]))
Reasoning about list equality
Two empty lists are equal; if one is empty and the other is not, they are not equal.
If both are nonempty, then their first elements must be equal, and their rests must be equal.
The natural recursion in this case is
(list =? (rest lst1) (rest lst2))

Review: Processing two lists at different rates
If the two lists lst1, lst2 being consumed are of different lengths, all four possibilities for their being empty/nonempty are possible:
(and (empty? lst1) (empty? lst2))
(and (empty? lst1) (cons? lst2))
(and (cons? lst1) (empty? lst2))
(and (cons? lst1) (cons? lst2))

Exactly one of these is true, but all must be tested in the template.

Group Problem - intersection
Write a function intersection that consumes two sorted (listof Nat) with such that an element appears at most once in a list (however, it could appear in both lists). The function produces a sorted list that is the intersection of the two lists. Provide a contract and purpose.
(check-expect (intersection '(2 4 6 8) '(1 2 3 4)) '(2 4))
(check-expect (intersection '(1 3 5 7) '(2 4 6 8)) '())
Clicker Question - List Translation
Given this list:
(list (list) 'cons (list (list 2 'green) 3))

Which of the following is equivalent to the given list?
A `(empty cons (list 2 'green) 3)
B `(empty cons (2 'green) 3)
C `(empty cons (list (list 2 'green) 3))
D `(( cons ((2 green) 3))
E `(() 'cons ((2 'green) 3))

Limitations of quoted lists:
• All identifiers become symbols, even special ones like true, false, and empty.
  `(empty true false) => (list 'empty 'true 'false)
• Function names also become symbols, so a quoted list cannot contain values obtained from applying functions or special forms (including structure values).
  `((make-posn 0 0)) => (list (list 'make-posn 0 0))

Group Problem - Remove Firsts
Write a function called remove-firsts that consumes a (listof (listof Any)). It should produce a new (listof (listof Any)) with the first item removed from each of the sublists. Note that if a list in the (listof (listof Any)) is empty, then no item should be removed from it. Do not use helper functions. Include the contract, examples and tests.

For testing you can include the following test:
(check-expect (remove-firsts '((1 2 3) (2 3) (1) ()))
'(2 3) (3) () ()))
Review: Dictionaries

You know dictionaries as books in which you look up a word and get a definition or a translation.

More generally, a dictionary contains a number of keys, each with an associated value.

Example: the telephone directory. Keys are names, and values are telephone numbers.

Example: your seat assignment for midterms. Keys are userids, and values are seat locations.

More generally, any two-column table can be viewed as a dictionary.

Example: your report card, where the first column contains courses, and the second column contains corresponding marks.

What operations might we wish to perform on dictionaries?

- **lookup**: given key, produce corresponding value
- **add**: add a (key,value) pair to the dictionary
- **remove**: given key, remove it and associated value

Group Problem - Class-Average

Given the data definition below, write a template function for a Class:

```scheme
;; A Student is (list Str (listof Num))
;; Str is a student id
;; Num is an assignment grade which is in the range [0, 100]

;; A Class is one of
;; empty
;; (cons Student Class)

;; Class is an Association List with Student entries
```
Group Problem - Class-Average

Given below is the data definition of Averaged-Class

;; An Averaged-Student is (list Str (anyof Num 'none))
;; Str is a student id
;; value (anyof Num 'none) is an averaged assignment grade which
;; is in range [0, 100] or 'none if student has no grades

;; An Averaged-Class is one of
;; empty
;; (cons Averaged-Student Averaged-Class)

;; Averaged-Class is an Association List with Averaged-Student entries

Group Problem - Class-Average

Write a function class-average which consumes a Class and produces an Averaged-Class where the grade for each student is the average of their grades in the consumed Class. Include only the contract for the Design Recipe.

;; Example:
(check-expect (class-average `(("stud1" (2 3 4)) ("stud2" ())))
  `(("stud1" 3) ("stud2" none)))