This tutorial will cover the following topics:

- Traversing binary search trees.
- Using local definitions to avoid repetition of common subexpressions, to improve readability of expressions, and to improve efficiency of code.
Question 1: Counting Keys in Binary Search Trees
In Module 11 (Slide 21), you learned about binary search trees. You should use the data definitions for BST and Node defined on Slide 22 when solving this problem:

;; A Binary Search Tree (BST) is one of:
;; * empty
;; * a Node

(define-struct node (key left right))
;; A Node is a (make-node Nat BST BST)
;; Requires: key > every key in left BST
;; key < every key in right BST

Part A: Write templates for processing a BST and a Node.
Part B: Write a function (count-range bst low high) which consumes a binary search tree bst and two natural numbers low and high. The function should produce the number of keys in the binary tree whose value is at least low and at most high. *You must solve this problem efficiently* (i.e., without searching paths that cannot contain a valid key).
;; Examples:
(define example-tree (make-node 5
  (make-node 1 empty empty)
  (make-node 7
    (make-node 6 empty empty)
    (make-node 14 empty empty)))))
(check-expect (count-range example-tree 6 7) 2)
(check-expect (count-range example-tree 0 100) 5)
Question 2: Below Average
After the recent midterm, you’ve obtained a list of marks for each student in CS 135. You’d like to know how which students scored below the average.

Write a function (below-average marks) which consumes a list of marks (numbers ranging from 0.0 to 100.0) and returns a new list containing only the marks that were strictly below average, in the same order as they appeared in the original list.

Important: You may only use local helper functions in your solution and must compute the average only once!

;; Example:
(check-expect (below-average (list 60 80 40 95 100)) (list 60 40))
Local Definitions
The Intermediate language provides the special form local, which contains a series of local definitions plus an expression using them.

(local [(define x_1 exp_1) ... (define x_n exp_n)] bodyexp)

What is the purpose of local?
- To make expressions more readable.
- To avoid computing the same thing multiple times (this will help us compute the average only once!)
- To make code more efficient.