Assignment Guidelines.

- This assignment covers material in Modules 4 and 5.
- Submission details:
  - Solutions to these questions must be placed in files a4q1.rkt, a4q2.rkt, a4q3.rkt, and a4q4.rkt, respectively, and must be completed using Racket.
  - Unless otherwise indicated in the question you may use only the built-in functions and special forms introduced in the lecture slides from CS115 up to and including the modules covered by this assignment. A list of functions described in each module of the lecture slides can be found at [https://www.student.cs.uwaterloo.ca/~cs115/built_in](https://www.student.cs.uwaterloo.ca/~cs115/built_in)
  - Download the interface file from the course Web page to ensure that all function names are spelled correctly and each function has the correct number and order of parameters.
  - All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
  - Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
  - For full style marks, your program must follow the CS115 Style Guide.
  - Be sure to review the Academic Integrity policy on the Assignments page.
  - For the design recipe, helper functions only require a purpose, a contract and an example.
- Restrictions:
  - Unless the question specifically describes exceptions, you are restricted to using the functions and special forms covered in or before Modules 4 and 5.
  - Read each question carefully for additional restrictions.
- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.

1. Max and min.

Write two functions:

1. (my-max L) consumes a non-empty (listof Num) and returns the largest value in the list.
2. (my-min L) consumes a non-empty (listof Num) and returns the smallest value in the list.

```
(my-max (list 3 12 42 37 3 5)) => 42
(my-min (list -3 -12 -42 -37 -3 -5)) => -3
(my-min (list 3 12 42 37 3 5)) => 3
```

Do not use the built-in `max`, `min`, or `sort` functions.
2. **Data Classification.**

How can we decide if something is “large” or “small”? If we have a single sample, there is nothing to compare it to. To classify items we need to compare them to a population of samples.

Recall the _mean_ and _standard deviation_ from an earlier assignment.

We will define something as “large” (“L”) relative to a population if it is more than one standard deviation above the mean, “small” (“S”) if it is more than one standard deviation below the mean, and “medium” (“M”) if it is within one standard deviation of the mean.

**Exercise**

Write a function `(classify L)` that consumes a `(listof Num)` and returns a list indicating if each item is "L", "S", or "M".

For example, for `(list 5 5 5 4 5 5 6 5)`, the mean is 5 and the standard deviation is 0.5. So

```
(classify (list 5 5 5 4 5 5 6 5)) => (list "M" "M" "M" "S" "M" "M" "L" "M")
```

```
(classify (list 1 2 3 4 5 6 7 8 9 10)) => (list "S" "S" "M" "M" "M" "M" "M" "M" "L" "L")
```

**Hint**

You may use the `mean` and `std` functions developed on a previous assignment: either your own code, or the code from the posted solution.

3. **Counting Things.**

**Exercise**

Write a function `(count item L)` that consumes an Any and a `(listof Any)` and returns a Nat representing the number of times item appears in L.

```
(count "sheep" (list 3 "sheep" "goat" "sheep" 56)) => 2
```

**Hint**

The predicate `(equal? a b)` consumes two Any, and returns #true if the two values are the same. (Recall `(= a b)` only consumes two Num.)

4. **Prime Numbers.** A Nat is prime if it has exactly two divisors: 1 and itself.

**Exercise**

(a) **Identifying.** Write a function `prime?` that consumes a Nat, and returns #true if the Nat is prime, and #false otherwise.

```
(prime? 5) => #true
```

```
(prime? 6) => #false
```

**Hint**

Use `range`, then create a list of divisors.

(b) **Listing.** Write a function `(primes n)` that consumes a Nat and returns a list of all prime numbers less than n.

```
(primes 23) => (list 2 3 5 7 11 13 17 19)
```