Assignment Guidelines.

• This assignment covers material in Module 10.

• Submission details:
  – Solutions to these questions must be placed in files a10q1.rkt, a10q2.rkt, a10q3.rkt, and a10q4.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
  – 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:
  – Read each question carefully for additional restrictions.

  ! Do not compute any value more than once. For example, if \( n \) is the length of \( L \), the following code computes \( \text{length} \) \( n \) times:

  ```racket
  (define (addlen L)
    (local [(define (add-len x) (+ x (length L)))]
      (map add-len L)))
  
  Do not do this kind of thing; instead, use \texttt{local} constants.
  ```

  ! Do not use \texttt{lambda} on this assignment. Use \texttt{local} helper functions instead.

  ! Do not write any non-\texttt{local} helper functions on this assignment.

• 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. Local Lists.

Write a function `(add-max-min L)` that consumes a non-empty `(listof Int)`, and adds the smallest and largest value to each value in `L`.

For example, in `(list 2 3 5 7)`, the smallest value is 2, and the largest is 7, so we add `2 + 7 = 9` to each value, and return `(list 11 12 14 16)`.

2. Zero-padding. You may use the built-in function `(replicate n c)` in this question.

```
(replicate 4 "0") => "0000"
```

Write a function `(zero-pad L)` that consumes a `(listof Nat)` and returns a `(listof Str)` where each value has been converted to a `Str`, then enough zeroes have been added so all the values are of equal length.

```
(zero-pad (list 6 1245 42)) => (list "0006" "1245" "0042")
```

Hint: Recall the `number->string` function.

3. z-score. The z-score is the number of standard deviations a value is from the mean.

```
Write a function `(z-scores L)` that consumes a `(listof Num)` and returns a `(listof Num)` representing the z-scores of the values in the list.

For example, the mean of `(list 5 5 8 8 8 8 11 11)` is 8. The standard deviation is:

\[
\sqrt{\frac{(5-8)^2 + (5-8)^2 + (5-5)^2 + (5-5)^2 + (5-5)^2 + (11-5)^2 + (11-5)^2}{9}} = \sqrt{\frac{36}{9}} = \sqrt{4} = 2
\]

Since the mean is 8 and the standard deviation is 2, the z-score of the 5s is −1.5. So
```
(z-scores (list 5 5 8 8 8 8 11 11)) => (list -1.5 -1.5 0 0 0 0 1.5 1.5)
```

Further examples:
```
(z-scores (list 8 4 8 4)) => (list 1 -1 1 -1), since the mean is 6 and the standard deviation is 2.
```

Hint: Create two local constants: the mean, and the standard deviation.

You may assume the standard deviation of the values is non-zero (include this as a requirement for your function). That is, it is OK if your function does not work for a list containing only one value, such as `(list 8 8 8 8)`.