Assignment Guidelines:

- Solutions for Questions 1–3 are expected to follow the requirements of the Style Guide (https://www.student.cs.uwaterloo.ca/~cs115/coursenotes1/styleguide.pdf). This includes all relevant design recipe elements, proper use of constants, and proper use of helper functions.
- Submission details:
  o Solutions to these questions must be placed in files `a02q1.rkt`, `a02q2.rkt`, `a02q3.rkt`, respectively, and must be completed in Racket. The results for Question 4 will automatically be recorded.
  o All solutions must be submitted through MarkUs. Solutions will **not** be accepted through email.
  o For Questions 1–3 , verify your basic test results using MarkUs to ensure that your files were submitted properly and are readable on MarkUs. *Note, however, that passing the basic tests does not guarantee that you will pass all our correctness tests.*
- 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.
- Restrictions:
  o You may only use the built-in functions and special forms introduced in the lecture slides in Modules 01 and 02. A list of these functions can be found on the Assignments web page: https://www.student.cs.uwaterloo.ca/~cs115/#allowed In particular you are not allowed to use `cond` in your solutions for this assignment.
  o Read each question carefully to see if any additional restrictions apply.
  o Test data for correctness tests will always meet the stated assumptions for consumed values.
- 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.

Language level: Beginning Student
Coverage: Module 2

Question 1: Complete the Racket function **`trim`** that consumes a string (**`s`**) and two natural numbers (**`front`** and **`back`**), and produces a substring formed by removing the first **`front`** characters of **`s`** and removing the last **`back`** characters of **`s`**.

For example, **`(trim "abcdefghi" 2 3) => "cdef"`**

Each of the values of **`front`** and **`back`** must be less than or equal to the number of characters in **`s`**. If the sum of **`front`** and **`back`** is greater than or equal to the number of characters in **`s`**, then the function should produce the empty string.

For example, **`(trim "abcdef" 3 4) => ""`**

Question 2: In statistics, the *mean* of a set of numbers is calculated by adding the numbers together and dividing the sum by the size of the set. The *variance* is a measurement of the distance the individual values are from the mean. The variance of a set of numbers is calculated by finding the mean of the squares of the differences between each of number and the mean of the set of numbers. For example, if you have three numbers a, b, c, and the mean of these numbers is m, then the variance is:

$$\frac{(a - m)^2 \; + \; (b - m)^2 + (c - m)^2}{3}$$

Complete the Racket function **`variance`** that consumes five numbers (**`n1`**, **`n2`**, **`n3`**, **`n4`**, and **`n5`**) and produces the variance of those numbers.

For example, **`(variance 2 3 2.5 3 2) => 0.2`**

Question 3: In computing, data is represented by a sequence of 0s and 1s known as bits. When transmitting data, errors may be introduced. One way to do error checking is to break up the sequence of bits into blocks and use a *parity bit* at the end of each block. For example, we can add a parity bit (a 0 or 1) to the end of each block that guarantees that the extended block has an odd number of 1s. After the data is transmitted, the parity of each extended block can be checked to see that there is an odd number of 1s. If there is not, we know there was an error.

Complete the Racket function **`add-parity-bit`** that consumes a string composed of seven bits and produces a string with 8 bits, with a correct parity bit added to the end.

For example, **`(add-parity-bit "1100101") => "11001011"`**

Hints (correct solutions may not need to use these):

- 0 is an even number
- Any number, raised to the power of 0 is 1. In particular, $0^0 = 1$.
- Any number raised to the power of 1 is itself, i.e. $n^1 = n$.
- You may find the built-in functions **`string->number`** and **`number->string`** useful.

Question 4*:* For this question, you will perform step-by-step evaluations of Racket programs, by applying substitution rules until you either arrive at a final value or you cannot continue. You will use an online evaluation tool that we have created for this purpose. You do not need to hand in any files for this question.

To begin, visit this webpage

 https://www.student.cs.uwaterloo.ca/~cs115/stepping

Note that the use of https is important; that is, the system will not work if you omit the s. This link can also be found on the CS115 course website, under the Assignments heading.

You will need to authenticate yourself using your Quest/WatIAm ID and password. Once you are logged in, try the "Warm-Up questions" under "CS115 Assignment 2", in order to get used to the system. Note the "Show instructions" link at the bottom of each problem. Read the instructions before attempting a question! When you are ready, complete the four stepping problems in the "Assignment 2 questions" category, using the semantics given in class for Beginning Student. You can re-enter a step as many times as necessary until you get it right, so keep trying until you completely finish every question. All you have to do is complete the questions online – we will be recording your answers as you go, and there is no file to submit. The basic tests for this assignment will tell you whether or not we have a record of your completion of the stepper problems.

Note however that you are not done with a question until you see the message "Question complete!" You should see this once you have arrived at a final value and clicked on "simplest form" (or "Error", depending on the question). You should not use DrRacket's stepper to help you with this question for several reasons. First, as mentioned in class, DrRacket's evaluation rules are slightly different from the ones presented in class, but we need you to use the evaluation rules presented in class. Second, in an exam situation, you will not have DrRacket's stepper to help you, and there will definitely be step-by-step evaluation questions on at least one of the exams.