Assignment Guidelines:

- **Do not** use any conditional expressions (cond) on this assignment.
- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch, including helper functions.
- **For full marks, it is not sufficient to have a correct program.** Be sure to follow all the steps of the design recipe. Read the Style Guide carefully to ensure that you are following the proper conventions. *In addition, your solution must include the definition of constants and helper functions where appropriate.*
- 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.
- Read each question carefully for restrictions.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include references to the parameter names of your functions.
- 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.
- Do not send any code files by email to your instructors or tutors. Course staff will not accept it as an assignment submission. Course staff will not debug code emailed to them.
- You may post general assignment questions using the discussion groups on Waterloo LEARN. Choose Connect → Discussions. Read the guidelines for posting questions. Do NOT post any code as part of your questions.
- Check Markus and your basic test results to ensure that your files were properly submitted. In most cases, solutions that do not pass the basic tests will not receive any correctness marks.
- Read the course web page for more information on assignment policies and how to organize and submit your work. Follow the instructions in the Style Guide.
- Your solutions should be placed in files a2qY.rkt, where Y is a value from 1 to 3.

**Plagiarism: The following applies to all assignments in CS115.**

- All work in CS 115 is to be done individually. The penalty for plagiarism on assignments (first offense) is a mark of 0 on the assignment and a 5% reduction of the final grade, consistent with School of Computer Science policy. In addition, a letter detailing the offense is sent to the Associate Dean of Undergraduate Studies, meaning that subsequent offenses will carry more severe penalties, up to suspension or expulsion.
- To avoid inadvertently incurring this penalty, you should discuss assignment issues with other students only in a very broad and high-level fashion. Do not take notes during such discussions, and avoid looking at anyone else’s code, on screen or on paper. If you find yourself stuck, contact the ISA or instructor for help, instead of getting the solution from someone else. Do not consult other books, library materials, Internet sources, or solutions (yours or other people’s) from other courses or other terms.
- Be sure to read the Plagiarism section at: https://www.student.cs.uwaterloo.ca/~cs115/assignments
Language level: Beginning Student
Coverage: Modules 1 and 2

- Refer to the String documentation found at:

1. A group of University students and staff went to eat at a restaurant. The restaurant owner wants to compute the total cost of everything consumed by the group.

   - Food costs $10 for a student and $12 for a staff member.
   - A drink costs $7.50 for a staff member whereas students get a 30% discount off the staff price.
   - Dessert costs $5 for a student and $10 for a staff member.

They all had food, drink and dessert. Write a Racket function `dining-cost` that consumes two natural numbers `num-student` and `num-staff`, representing the number of students and staff members who went to the restaurant, and produces the total cost of everything consumed.

For example:
- `(dining-cost 2 2) => 99.5`
- `(dining-cost 10 5) => 350`

2. A string may be split into two substrings of equal lengths if the length of the string is an even integer. However, if the length of the string is odd, the length of the first half should be one less than the length of the second half. For example, the first half of "a" is "", and the second half is "a"; the first half of "computer" is "comp", and the second half is "uter"; and the first half of "computers" is "comp", and the second half is "uters".

Write a Racket function called `substring-lengths` that consumes a non-empty string `s` and produces a new string obtained by concatenating substrings of `s` and their lengths strictly as follows:
- the length of the first half of `s`,
- the first half of `s`,
- the length of the second half of `s`,
- the second half of `s`, and
- the length of `s`.

For example:
- `(substring-lengths "a") => "01a1"
- `(substring-lengths "computer") => "4comp4uter8"
- `(substring-lengths "computers") => "4comp5uters9"

Hint: you may need to use the quotient and/or number->string functions.
3. Write a Racket function called `string-insert` that consumes two strings `main` and `new`, and an integer `position`. The function produces another string by inserting `new` at index `position` in `main`. If `position` is less than 0, then `new` is inserted at the beginning of the string, and if `position` is greater than the length of `main`, then `new` is inserted at the end of the string.

   For example:
   - `(string-insert "sheep" "ba" 10) => "sheepba"
   - `(string-insert "sheep" "ba" 2) => "shbaeep"
   - `(string-insert "sheep" "ba" -20) => "basheep"

   Hint: you may need to use the functions `min` and/or `max`.

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: 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 webpage, under the Assignments heading.

   You will need to authenticate yourself using your Quest/WatIAM ID and password. Once you are logged in, try the “Warmup questions” under “CS 115 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.