
- **Academic Integrity Links**
  - [https://uwaterloo.ca/academic-integrity/basic-info](https://uwaterloo.ca/academic-integrity/basic-info)
  - [https://uwaterloo.ca/library/get-assignment-and-research-help/academic-integrity/academic-integrity-tutorial](https://uwaterloo.ca/library/get-assignment-and-research-help/academic-integrity/academic-integrity-tutorial)
- You must provide the data definition and template in your solutions only when the question specifically indicates they are required for compound data types described in the question. If you create any additional data types that are beyond the question description, your program file should include a data definition and a template for each additional data type.
- 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.
- You may want to include defined constants to help reduce the writing for the examples and tests.
- Unless otherwise indicated by the question, you may only use 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 on the Course Website, at [https://www.student.cs.uwaterloo.ca/~cs115/built_in](https://www.student.cs.uwaterloo.ca/~cs115/built_in)
- Use the design recipe when writing functions (and helper functions) from scratch.
- 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.
- 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.
- 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.
- 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.
- Any string or symbol values must exactly match the description in the questions. Any discrepancies in your solutions may lead to a severe loss of 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 a09qY.rkt, where Y is a value from 1 to 2.
Language Level: Intermediate Student with lambda
Coverage: Module 9 and 10

For question 1, you are not allowed to use recursion. Your solutions must use the abstract list functions (map, filter, foldr, build-list). All constants and helper functions must be local definitions.

1. Complete parts a,b,c using local to encapsulate constants and helper functions
   a. Complete a Racket function called largest-difference that consumes a non-empty list of numbers alon and produces the difference between the largest and smallest number in alon.
      For example:
      (largest-difference (list 9 4 8 2 1)) => 8
      (largest-difference (list -4 3.12 9.5 2 1)) => 13.5
   b. Complete a Racket function called average-length that consumes a non-empty list of strings alos and produces the average length of all the strings in alos.
      For example:
      (average-length (list “final” “CS” “assignment!”)) => 6
   c. Complete a Racket function called largest-posn that consumes a non-empty list of posns alop. The function produces the posn with the largest sum of its coordinates. If there is a tie, produce the posn that occurred first in the list.
      For example:
      (largest-posn (list (make-posn 4 8) (make-posn 3 1) (make-posn 3 6))) => (make-posn 4 8)

Complete parts d,e,f using lambda to create helper functions without local

   d. Complete a Racket function called small-numbers-count that consumes a non-empty list of numbers alon and a number threshold. The function produces how many numbers in alon are less than threshold.
      For example:
      (small-numbers-count (list 2 6 8 5 1 9) 5) => 2
   e. Complete a Racket function called build-matrix that consumes a natural number n greater than 0. The function produces a list of n lists where each list contains n copies of n.
      For example:
      (build-matrix 3) => (list (list 3 3 3) (list 3 3 3) (list 3 3 3))
      (build-matrix 5) => (list (list 5 5 5 5 5) (list 5 5 5 5 5) (list 5 5 5 5 5) (list 5 5 5 5 5) (list 5 5 5 5 5))
   f. Write a Racket function called only-vowels that consumes a list of strings alos containing only lower-case letters and produces a list of only the strings from alos that contain at least one vowel. A vowel is one of {“a”, “e”, “i”, “o”, “u”, “y”}.
      For example:
      (only-vowels (list “ybc” “def” “gh” “jkl”)) => (list “ybc” “def”)
2. A generic-leaf-labelled tree (GenLLT) is one of the following:

   ;; * empty
   ;; * (cons Num GenLLT)
   ;; * (cons Str GenLLT)
   ;; * (cons Sym GenLLT)
   ;; * (cons Bool GenLLT)
   ;; * (cons GenLLT GenLLT) where the first GenLLT is non-empty

Complete a Racket function called `num-str-diff` that consumes a GenLLT `agenllt`. The function produces the difference between the sum of all the numbers and the sum of the length of all the strings in `agenllt`.

For example:

   a. `(num-str-diff (list 6 'sadly "done" (list "CS115" 5)) => 2
   b. `(num-str-diff (list true 'the (list 1 2 (list "final" 5))
      "assignment" "question" false (list true 10 8 "but also" (list
      "the" "hardest" 4 3) 2 'question))) => -6
   c. `(num-str-diff empty) => 0