Assignment 06
Due: Wednesday, November 8 at 10am

• **Note that the interface file provided for this assignment does not include function headers.**
• You are responsible for ensuring that your functions match the description in the questions.
• In particular, you must use exactly the same function names specified by the question, and you must have the same number of parameters as described, and they must occur in the order specified in the question.
• Problems with the names of functions and number of parameters will be identified by the basic tests, so be sure to submit your solutions early in order to avoid losing all correctness marks for such errors.
• **Do not use reverse.**
• If your solution includes a wrapper function, you should provide the full design recipe, but wrapped functions require only the purpose and contract.
• 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 type that are beyond the question description, your program file should include a data definition and a template for each additional data type.
• If you include a template in your solution, the template should appear as comments.
• You may want to include defined constants to help reduce the writing for 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 may be found 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 structures are defined correctly.
• 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 forum on Waterloo LEARN. Choose Connect -> Discussions. Read the guidelines for posting questions. Do NOT post any code as part of your questions.
• 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.
• 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 descriptions 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 a06qY.rkt, where Y is a value from 1 to 3.
Language level: Beginning Student with List Abbreviations
Coverage: Module 6

Question 1:
In this assignment, you are required to create your own parameter names for the function and the parameters must occur in the order specified in the question.

Using structural recursion, write a Racket function called even-up-odd-down that consumes a natural number (say, n) and an integer (say, current) and produces a list that contains:
- n even integers greater than or equal to current arranged in ascending order, and
- n odd integers greater than or equal to current arranged in descending order.

For example:
(even-up-odd-down 2 5) => (list 6 8 7 5)
(even-up-odd-down 3 4) => (list 4 6 8 9 7 5)
(even-up-odd-down 0 6) => empty
(even-up-odd-down 2 -5) => (list -4 -2 -3 -5)

Use the following structures and data definitions to complete questions 2 and 3:

(define-struct fullname (first last))
;; A Fullname is a (make-fullname Str Str)

(define-struct student (id name grades))
;; A Student is a (make-student Nat Fullname (listof Num))
;; requires:
;;   id is a unique natural number representing the student id
;;   name is the fullname of the student
;;   grades is a non-empty list of student grades between 0 and 100 inclusive

The constants below are used in the examples provided for questions 2 and 3.

(define st1 (make-student 1 (make-fullname "Joe" "Trudeau")
             (list 100 93.2 71.4 84.6 78.45)))
(define st2 (make-student 2 (make-fullname "Keaton" "Abass")
             (list 50 60 43 22.5 32.8)))
(define st3 (make-student 3 (make-fullname "Tolu" "Mandela")
             (list 50 60 70 80 62 51 40)))
(define st4 (make-student 4 (make-fullname "Xi" "Mandela")
             (list 2.5 5.5))
(define st5 (make-student 5 (make-fullname "Gupta" "Merkel")
             (list 65 62.5 69.3 88.54 61.2 54 74.3))
Question 2:
In this assignment, you are required to create your own parameter names for the function and the parameters must occur in the order specified in the question.

Using structural recursion, write a Racket function called sort-students that consumes a list of students and produces a new list of students sorted in a case-insensitive alphabetic order of the last name. If students have the same last name, sort by their first names, also case-insensitive. If there is a duplicate first and last names, sort by their id.

*Hints:* You may modify the insert and sort functions in Module 06, Slides 41-47. You may also use the string functions string-ci<?, string-ci?>, string-ci<=? and string-ci=>? in your solution.

For example:

```
(sort-students (list st1 st3 st5)) => (list st3 st5 st1)
(sort-students (list st1 st2 st3 st4 st5)) =>
    (list st2 st3 st4 st5 st1)
```

Question 3:
In this assignment, you are required to create your own parameter names for the function and the parameters must occur in the order specified in the question.

Using structural recursion, write a Racket function called select-students that consumes a list of student structure and two natural numbers (between 0 and 100 inclusive). The function produces a list of student ids whose average grade falls between the two natural numbers inclusive. If the list is empty, the function should produce empty. The student ids in the list produced should appear in the order they were in the original list. The first natural number may be less than, equal, or greater than the second one.

For example:

```
(select-students (list st1 st2 st3 st4 st5) 80 50) => (list 3 5)
(select-students (list st5 st4 st3 st2 st1) 50 80) => (list 5 3)
```