Deadline: Wednesday, February 12, 2020 at 10:00am on MarkUs

Language level: Beginning Student

Coverage: Module 04

Files to submit: `a04q1.rkt, a04q2.rkt, a04q3.rkt`

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:
  - All solutions must be submitted through MarkUs. Solutions will not be accepted through email.
  - 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: passing the basic tests does not guarantee that you will pass all our correctness tests!
- Download the interface files from the course web page to ensure that all function names are spelled correctly, and that each function has the correct number and order of parameters.
- **Restrictions:**
  - Unless specifically allowed in the description of the question, you may only use the built-in functions and special forms introduced in the lecture slides in Modules 01-04. For details, see https://www.student.cs.uwaterloo.ca/~cs115/#allowed
  - Read each question carefully to see if any additional restrictions apply.
  - Test data for correctness tests will always meet the stated assumptions for consumed values.
  - **Reminder:** Do NOT copy/paste text from the Assignment PDF to your definitions window. This can cause errors!
- 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.

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 affected questions 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.

Read more course policies at: https://www.student.cs.uwaterloo.ca/~cs115/#policies

1. **List Summation and Divisor Checking**

   Write a Racket predicate `divides-sum?` which consumes a positive integer `divisor` and a list of integers `intz`. The predicate will produce `true` if the sum of all of the elements of `intz` is a multiple of `divisor`, and `false` otherwise.

   Note that the empty list will be considered to have a sum of 0 and therefore should always return `true` when consumed as `intz`, regardless of the `divisor`.

   Sample examples:

   - `(divides-sum? 5 (cons 5 (cons -2 (cons 2 (cons 0 empty)))))` ⇒ `true`
   - `(divides-sum? 99999 empty)` ⇒ `true`
   - `(divides-sum? 7 (cons 1 (cons 2 (cons 3 (cons 4 (cons 5 empty)))))` ⇒ `false`

   Place your definitions into the file **`a04q1.rkt`**.

2. **Red v.s. Blue**

   Write a Racket function `red-vs-blue` which consumes a list `ramble`, and produces a string according to the following rules:

   - produces `"red"` if there are more occurrences of the string `"red"` in `ramble` than `"blue"`,
   - produces `"blue"` if there are more occurrences of the string `"blue"` in `ramble` than `"red"`, and
   - produces `"tie"` if there are an equal number of occurrences of `"red"` and `"blue"` in `ramble` (this includes lists which contain neither string).

   Sample examples:

   - `(red-vs-blue (cons "red" (cons "blue" (cons 5/7 (cons "blue" empty)))))`
     ⇒ `"blue"`
   - `(red-vs-blue empty)` ⇒ `"tie"`
   - `(red-vs-blue (cons 3 (cons 1 (cons 4 (cons 1 (cons 5 (cons 9 empty)))))))` ⇒ `"tie"`
   - `(red-vs-blue (cons "red" (cons "green" empty)))` ⇒ `"red"`

   Place your definitions into the file **`a04q2.rkt`**.

3. **Clicker Marks**

You would like to calculate your clicker mark, given a list of scores for each clicker question. The first step is to determine the number of questions you answered correctly, the number of questions you answered incorrectly, and the number of questions you did not answer. As you should know, each clicker question will result in either a score of 2 (answered correctly), 1 (answered incorrectly) or 0 (not answered).

Write the Racket function `count210` that consumes a list, `scores`, containing only the integers 0, 1 or 2. The function `count210` produces a list of size 3. The first element of the list produced is the number of 2s, the second element of the list produced is the number of 1s, and the third element of the list produced is the number of 0s in the consumed list. For example:

```
(count210 (cons 2 (cons 2 (cons 0 (cons 1 (cons 2 (cons 0 empty)))))))
 ⇒  (cons 3 (cons 1 (cons 2 empty)))

(count210 empty)  ⇒  (cons 0 (cons 0 (cons 0 empty)))
```

Place your definitions into the file **`a04q3.rkt`**.