Deadline: Wednesday, February 5, 2020 at 10:00am on MarkUs

Language level: Beginning Student

Coverage: Module 01-03

Files to submit: `a03q1.rkt, a03q2.rkt, a03q3.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 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:
  - 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 Module 01-03. 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.
- 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 offence is sent to the Associate Dean of Undergraduate Studies, meaning that subsequent offences 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. **Duplicate Detection**

   Note: you are not allowed to use `cond` for this question.

   Write a Racket predicate `duplicates?` that consumes four values `v1`, `v2`, `v3` and `v4` of any type.

   This predicate produces `true` if at least two values are the same, and `false` otherwise.

   For example,

   - `(duplicates? 1 2 3 1)` ⇒ `true`
   - `(duplicates? "A" "A" "A" false)` ⇒ `true`
   - `(duplicates? true false "true" "false")` ⇒ `false`

   Place your definitions into the file `a03q1.rkt`.

2. **Check And Extract**

   Write a Racket function `check-and-extract` which consumes a string, `type-value`, with the format `"<type>:<text>"` where

   - `<type>` is replaced by an actual type, one of `Str`, `Num`, `Int`.
   - `<text>` replaced by some text; the text might be empty.

   The function produces either a string, a number, or `false` depending on the `<type>` that appears at the beginning of `type-value`. Specifically, if the format of `type-value` is:

   - `"Str:<text>"`
     - The function should produce the text itself. For example:
       `(check-and-extract "Str:Hello World!")` ⇒ `"Hello World!"`
       `(check-and-extract "Str:3.14159")` ⇒ `"3.14159"`
   - `"Num:<text>"`
     - The function first verifies if the text is an actual number If so, produce the number; `false` otherwise. For example:
       `(check-and-extract "Num:3.14")` ⇒ `3.14`
       Note: not the string `"3.14"`*!*
       `(check-and-extract "Num: 3.14")` ⇒ `false`
       `(check-and-extract "Num:0 is a number. I promise.")` ⇒ `false`
       The string `" 3.14"` *(note the space in front of the 3) and* `"0 is a number. I promise."` *cannot be converted to a number.*
   - `"Int:<text>"`
     - The function first verifies if the text is an actual integer. If so, produce the integer; `false` otherwise. For example:
       `(check-and-extract "Int:-5")` ⇒ `-5`
       `(check-and-extract "Int:4/5")` ⇒ `false`
       `(check-and-extract "Int:forty-two")` ⇒ `false`

For this question, you might want to use the built-in function `string->number`, documented in the string documentation (https://www.student.cs.uwaterloo.ca/~cs115/resources/strings.pdf).

Place your definitions into the file `a03q2.rkt`.

3. **Compare Three Numbers**

Write a Racket function `compare-three` that consumes three numbers `a`, `b`, and `c` (in this order) and produces a string. The string includes only the letters `"a"`, `"b"`, and `"c"` along with `"="` and/or `"<"` to indicate the ranking of the numbers consumed. The ranking is indicated as follows:

- If all three numbers are equal, produce `"a=b=c"`.
- If exactly two of three numbers are equal, then the third number is either smaller or larger than the other two. For example, assume $a = b$:
  - if c is less than a and b produce `"c<a=b"`,
  - otherwise produce `"a=b<c"`.
- If none of the numbers are equal, produce `a`, `b`, `c` in the order from least to greatest, separated by a single `<`. For example, if $c < b < a$, produce `"c<b<a"`.

When numbers are equal, the string must show the letters in alphabetical order from left to right. For example, `"c=b=a"` and `"b<c=a"` will be considered incorrect, even if they show correct ranking mathematically.

For example,

- `(compare-three 1 2 3)` ⇒ `"a<b<c"`
- `(compare-three 1.7 12 0.5)` ⇒ `"c<a<b"`
- `(compare-three 1/2 0.5 3)` ⇒ `"a=b<c"`
- `(compare-three 1/4 0 0.25)` ⇒ `"b<a=c"`

Note: The string produced must match the format exactly. There are no spaces between any characters in the produced string, and all letters are in lower case. Adding in extra spaces or using upper case letters in your produced value will result in a mark of 0 for correctness.

Place your definitions into the file `a03q3.rkt`.

4. **Tracing**

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 3", 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 3 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.