CS115 **Assignment 03** Fall 2019

Due: Wednesday, October 2 @ 10:00 AM (No late submissions)

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:
  - Solutions to these questions must be placed in files **`a03q1.rkt`**, **`a03q2.rkt`**, **`a03q3.rkt`**, respectively, and must be completed in Racket. The results for Question 4 will automatically be recorded. When the stepping questions have been completed, MarkUs should display public tests results for Question 4.
  - 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, however, that 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:
  - You may only use the built-in functions and special forms introduced in the lecture slides in Modules 01, 02, and 03. A list of these functions can be found on the Assignments web page: 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 question and 5 marks off 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.

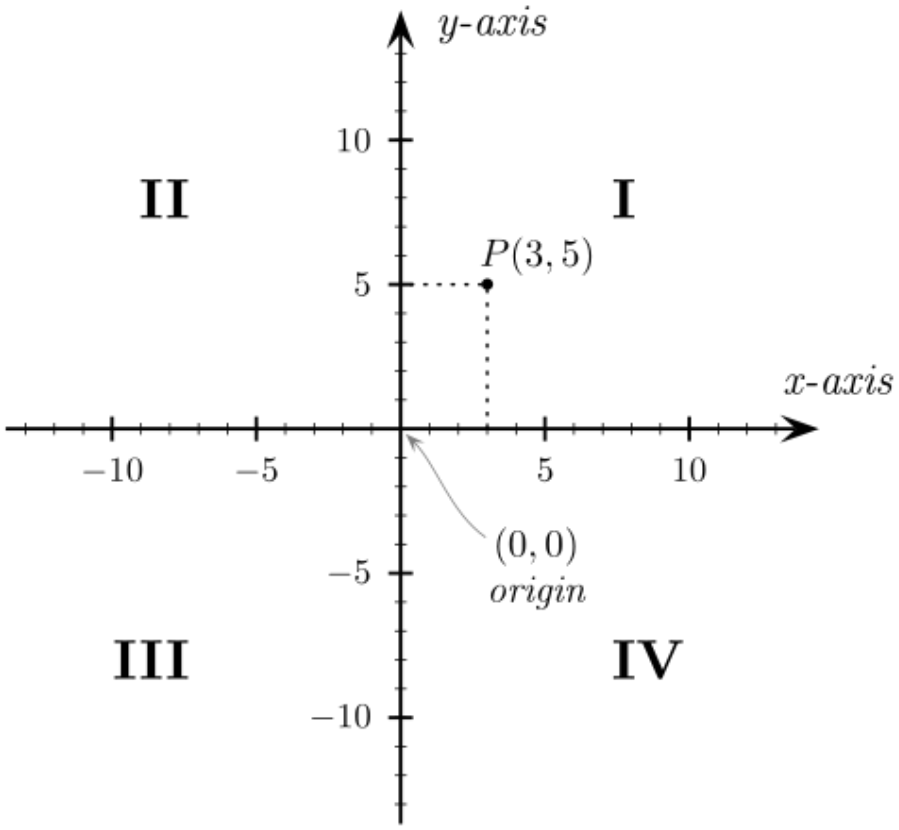
To avoid inadvertently incurring this penalty, you should discuss assignment issues with other students only in a very broad and high-level fashion. Do not take notes during such discussions, and avoid looking at anyone else's code, on screen or on paper. If you find yourself stuck, contact the ISA or 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

Language level: Beginning Student
Coverage: Module 3

Due: Wednesday, October 2 @ 10:00 AM (No late submissions)

Question 1: You may recall the labelling of the quadrants in the Euclidean plane as follows:



(image from Wikipedia https://en.wikipedia.org/wiki/Quadrant_(plane_geometry))

Complete the Racket function **`quadrant`** that consumes the **`x`** and **`y`** position of a point (both **`x`** and **`y`** are numeric values), and produces the quadrant (as an integer **`1`**, **`2`**, **`3`**, or **`4`**) the point (**`x`**, **`y`**) is located in; if the point is at the origin, produce **`0`**; If the point is on the x-axis, produce **`-1`**, and if the point is on the y-axis, produce **`-2`**.

For example:

- **`(quadrant 3 5) => 1`**
- **`(quadrant -5 -2/3) => 3`**
- **`(quadrant pi (- (sqrt 2))) => 4`**
- **`(quadrant 0 0) => 0`**
- **`(quadrant 0 9) => -2`**

Due: Wednesday, October 2 @ 10:00 AM (No late submissions)

Question 2: Do not use cond on this question.

A sequence is *strictly increasing* if each term in the sequence is larger than the term before it.

A sequence is *strictly decreasing* if each term in the sequence is smaller than the term before it.

The *parity* of an integer is even if its remainder is 0 when divided by 2, and is odd if its remainder is 1 when divided by 2. Note that there are two built-in Racket predicates, **`even?`** and **`odd?`**, that will be helpful.

An integer sequence has *alternating parity* if every term has the opposite parity of the term immediately preceding (if there is a term preceding it) and immediately following it (if there is a term following it). For example, the sequence 1 2 3 has alternating parity (odd even odd), but the sequence 2 8 19 (even even odd) does not have alternating parity.

An integer sequence has *uniform parity* if each term has the same parity.

Complete the Racket predicate called **`strict-parity-seq?`** which consumes 4 integers **`n1`**, **`n2`**, **`n3`**, and **`n4`**, and produces **`true`** if

- the sequence is strictly increasing or strictly decreasing, and
- the sequence has alternating parity or uniform parity,

and produces **`false`** otherwise.

For example:

- `(strict-parity-seq? 1 2 3 4) => true`
- `(strict-parity-seq? 4 2 1 3) => false`
- `(strict-parity-seq? -1 -5 -9 -13) => true`
- `(strict-parity-seq? 1 1 1 1) => false`
- `(strict-parity-seq? 4 2 -1 0) => false`

Question 3:

Part (a): Complete the Racket predicate **`same?`** that consumes three parameters **`a`**, **`b`**, and **`c`** (all three parameters can be any type), and produces **`true`** if all three parameters are the same, and **`false`** otherwise. For example:

- `(same? "a" 19 "a") => false`
- `(same? "c" "c" "c") => true`

Part (b): Complete the Racket function **`tic-tac-toe-win`** that consumes 9 strings **`s1`**, **`s2`**, **`s3`**, **`s4`**, **`s5`**, **`s6`**, **`s7`**, **`s8`**, and **`s9`**, and produces either **`"U"`** (if the game is unfinished), **`"X"`** (if the game ends with X as the winner), **`"O"`** (if the game ends with O as the winner), or **`"T"`** (if the game ends in a tie). For this question, if the function was

called with **`(tic-tac-toe-win s1 s2 s3 s4 s5 s6 s7 s8 s9)`**, the 9 parameters represent the contents of the tic-tac-toe board as shown below:

`s1`	**`s2`**	**`s3`**
`s4`	**`s5`**	**`s6`**
`s7`	**`s8`**	**`s9`**

and each of the values **`s1`**, **`s2`**, …, **`s9`** will be one of **`"X"`**, **`"O"`** or **`" "`** (a single space character). To win the game of tic-tac-toe, there must be three identical symbols (**`"X"`** or **`"O"`**) in the same row, or the same column, or through one of the two diagonals. Note that if there is a winner, that winner will be unique (i.e., no input will have both X and O winning at the same time). If there is no winner, and there are no unfilled board positions, the game ends in a tie. If there is no winner and there are unfilled board positions, the board is unfinished. For example:

- **`(tic-tac-toe-win "X" "O" "O"`**
  `"O" "O" "X"`
  `"O" "X" " ") => "O"`
- **`(tic-tac-toe-win " " "O" " "`**
  `"O" "X" "O"`
  `"O" "X" "O") => "U"`
- **`(tic-tac-toe-win "X" "X" "X"`**
  `"O" "X" "O"`
  `"O" "O" " ") => "X"`
- **`(tic-tac-toe-win "X" "O" "X"`**
  `"O" "O" "X"`
  `"X" "X" "O") => "T"`

You may want to use your solution to part (a) as a helper function for part (b), though you are not required to do so.

Question 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 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.

Due: Wednesday, October 2 @ 10:00 AM (No late submissions)

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.