Assignment: 3
Due: Tuesday, January 24, 9:00 pm
Language level: Beginning Student
Files to submit: lines.rkt, cards.rkt, triangle.rkt (bonus)
Warmup exercises: 6.3.2, 6.4.1, 7.1.2
Practise exercises: 6.3.3, 6.4.2, 6.4.3, 6.5.2, 7.1.3, 7.5.1, 7.5.2, 7.5.3

• Coverage: until M4-16
• Policies from Assignment 2 carry forward.
• Your solutions must be entirely your own work.
• Solutions will be marked for both correctness and good style.
• Good style includes qualities such as meaningful names for identifiers, clear and consistent
  indentation, appropriate use of helper functions, and documentation (the design recipe).
• For this and all subsequent assignments, you should include the design recipe as dis-
  cussed in class (unless otherwise noted, as in question 1).
• Test data for all questions will always meet the stated assumptions for consumed values.
• You must use check-expect for both examples and tests of functions that produce exact values.
  You must use check-within for examples and tests of functions that produce inexact values.
• You may use the cond special form. You are not allowed to use if in any of your solutions.
• It is very important that the function names match ours. You must use the basic tests to be sure. In most cases, solutions that do not pass the basic tests will not receive any correctness marks. The names of the functions must be written exactly. The names of the parameters are up to you, but should be meaningful. The order and meaning of the parameters are carefully specified in each problem.
• Any string or symbol constant values must exactly match the descriptions in the questions. Any discrepancies in your solutions may lead to a severe loss of correctness marks. Basic tests results will catch many, but not necessarily all of these types of errors.
• Since each file you submit will contain more than one function, it is very important that the code runs. If your code does not run then none of the functions can be tested for correctness.
• 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.
• You may use examples from the problem description in your own solutions.
Here are the assignment questions you need to submit.

1. In 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.

   To begin, visit this web page:

   [https://www.student.cs.uwaterloo.ca/~cs135/stepping](https://www.student.cs.uwaterloo.ca/~cs135/stepping)

**Note:** the use of `https` is important; that is, the system will not work if you omit the s. This link is also in the table of contents on the course web page.

   You will need to authenticate yourself using your Quest/WatIAm ID and password. Once you are logged in, try the questions in the “Warmup questions” category under “CS 135 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 six stepping problems in the “Assignment 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 are tracing the given expressions assuming that all function and constant definitions that appear above the expression exist. This means that you are not required to do any simplification of the constant definitions as part of the tracing.

   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.

**Note:** If you get stuck in the tracing question, **do not post to Piazza requesting the next step in the trace.** This is a violation of the academic integrity policy. Review the substitution rules carefully from Module 3 to try to solve the problem yourself. If you still cannot find your error, then you are encouraged to ask a question in person during office hours. As a last resort, you may make a private post to Piazza describing where you are stuck. Course staff will provide guidance directing you to the next step, but they will not give you the answer. Public post to Piazza related to question 1 of this assignment will be deleted.
2. Recall from high school geometry that two points, \((x_1,y_1)\) and \((x_2,y_2)\), define a line in a plane.
One standard equation for a line is \(y = mx + b\) where \(m\) is the slope, defined by
\[
m = (y_1 - y_2)/(x_1 - x_2)
\]
and \(b\) is the \(y\)-intercept (i.e., the point where the line crosses the \(y\)-axis).
This form of the equation of a line does not work well for vertical lines, however (since the slope is undefined, and there is no unique \(y\)-intercept). If the line is vertical, there is a unique \(x\)-intercept which does define the vertical line.
This question will be using the following structure and data definition.

\[(define-struct line (slope intercept))\]
;; A Line is a (make-line (anyof Num 'undefined) Num).

(a) Write a function \(\text{points->line}\) which consumes two distinct points (as \(Posn\) structures) and produces the corresponding \(Line\) which goes through these two points. Specifically, you should produce a \(Line\) structure with the slope set to the computed slope between the two points, and the intercept to be the \(y\)-intercept. Should your \(Line\) be vertical, you should set the slope of the line to be 'undefined and set the intercept to be the intercept on the \(x\)-axis.

(b) Write a function \(\text{perpendicular-line}\) that consumes a \(Posn\) representing a point and a \(Line\) in that order. Your function should produce a \(Line\) which goes through the given point and is perpendicular to the given \(Line\). Again, your consumed or produced \(Line\) may be vertical, and should be treated the same way as in part (b).

As a reminder, a perpendicular line has slope which is the negative-reciprocal to the original line (i.e., the negative-reciprocal of 2 is \(-\frac{1}{2}\)).

(c) Write a function \(\text{parallel/intersect}\) that consumes two distinct \(Line\) structures. Your function should produce \(true\) if the two consumed \(Lines\) are parallel, otherwise the function produces a \(Posn\) that represents the intersection point between the two consumed \(Lines\). For example \(\text{parallel/intersect (make-line 10 10) (make-line 8 4)}\) produces \(\text{(make-posn } -3 -20)\).

As a reminder, parallel lines are lines in a plane which do not meet; that is, two lines in a plane that do not intersect or touch each other at any point are said to be parallel.

Place your solutions in the file \(lines.rkt\).
3. A Card is defined by its suit and its rank.

The suit will be one of 'clubs, 'diamonds, 'hearts, or 'spades. The rank is an integer in the range 1 to 13, inclusive.

Place your solution to the following questions in the file cards.rkt. Note that your symbols must match the expected symbols exactly.

(a) Write a structure definition and an accompanying data definition for a Card. Use the field names as described above (with suit being the first field). As in the previous question, use the capitalization rules outlined in Module 04.

(b) Write a function better-card which consumes two Cards, and produces the Card which is the better of the two.

The better Card is either the Card with the better suit (the suits are increasing from 'clubs (worst), 'diamonds (second-worst), 'hearts (second-best), or 'spades (best)) or, if the Cards have the same suit, then the best Card is the Card with the largest/highest rank. If the two consumed Cards are the same (i.e., same suit and rank), return either one.

(c) Write the function hand-value which consumes three distinct Cards, and produces a symbol indicating the best hand-value of the given three Cards (where the arrangement/order of the three Cards can be altered). The hand-values are described in decreasing order (from best to worst) below:

- 'straight-flush: all three Cards are the same suit and their ranks are three consecutive integers;
- 'flush: all three Cards are the same suit;
- 'straight: the ranks of the three Cards are three consecutive integers;
- 'three-of-a-kind: the ranks of the three Cards are the same;
- 'pair: the ranks of two of the Cards are the same
- 'useless: none of the previous outcomes are satisfied.

This concludes the list of questions for which you need to submit solutions. Don’t forget to always check your email for the public test results after making a submission.
4. **BONUS QUESTION (5%)**

Structures are incredibly versatile. The fields of a structure can be any type and they can include other structures. Since functions in Racket can only produce a single result, structures can be used as a way to contain multiple data values within a single reference.

Recall that a `posn` structure represents a point on the Cartesian plane. Suppose you have another structure, that contains three `posn` structures defined as follows:

A *triangle* is a structure, of type `Triangle`, with three fields that appear in the following order:

- `point1`, a `posn` representing one vertex of the triangle.
- `point2`, a `posn` representing one vertex of the triangle.
- `point3`, a `posn` representing one vertex of the triangle.

Write a function called `triangle-area` that consumes a `Triangle` structure and produces a number representing the area of the shape if the three points actually form a triangle, and produces `false` if the three points are collinear (i.e. form a line) or if two or more of the points are the same. You may need to do some online research to find out how to calculate the area and/or how to determine if points are collinear.

Include all appropriate structure and data definitions.

As usual, this is an all-or-nothing type bonus question. No partial marks are awarded.

Place your solution for the bonus question in in the file `triangle.rkt`