- 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 can be found on LEARN.
- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch, including helper functions.
- 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.
- Read each question carefully for restrictions.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Download the interface file 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.
- 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.
- 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 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.
- 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 `a3qY.rkt`, where `Y` is a value from 1 to 2.
- Since each file you submit will contain more than one function, **it is very important that your code runs**. If your code does not run then none of the functions can be tested for correctness.

Language level: Beginning Student
Coverage: Module 3

This assignment makes extensive use of helper functions. You are told to write some helper functions explicitly, and you may choose to create others yourself. Break each problem down into smaller problems in order to decide which helper functions you will need to create.

1. **Bridge contracts: the Level and the Suit**
   For this question, you will submit the functions described in parts 1a and 1b in a single file.

   Bridge is a card game with four players, where the players are divided into two teams with two players each. After the cards are dealt, the players take part in an auction that determines the **contract** for the hand. The contract has two parts: a **level** and a **suit**. The level is a number between 1 and 7 inclusive, and the suit is one of clubs, diamonds, hearts, spades, or no trump. After the hand is played, if the team that won the auction satisfies their contract, then

they get points. The number of points they receive depends on the contract level and suit.

(a) Write a function called `basic-points` that consumes an integer between 1 and 7 inclusive (`level`), and a string (`suit`), which is one of the values `"clubs"`, `"diamonds"`, `"hearts"`, `"spades"` or `"NT"`, and produces the basic points for that contract. The basic points are calculated according to the following rules:
- minor suits (clubs and diamonds) are worth 20 points per level,
- major suits (hearts and spades) are worth 30 points per level, and
- no trump is worth 40 points for level 1, plus 30 points for all subsequent levels.

For example,
- `(basic-points 2 "clubs")` produces `40`,
- `(basic-points 4 "hearts")` produces `120`, and
- `(basic-points 3 "NT")` produces `100` (40 points for level 1 and $2 * 30$ for the remaining two levels).

(b) There are more rules that determine the actual points for a contract. The contract can be either **vulnerable** or **not vulnerable**. The contract may be **doubled** or **redoubled**. A contract at level 6 is called a **small slam**, and is worth bonus points. A contract at level 7 is called a **grand slam**, and is also worth bonus points. The full amount of points earned for a contract is calculated according to the following rules:
- a redouble can only happen after a double has been bid,
- if the contract is redoubled, it receives
  - 4 times the basic points, as they are described in part 1a and
  - a 100 point bonus,
- if the contract is doubled, but not redoubled, it receives
  - 2 times the basic points as they are described in part 1a, and
  - a 50 point bonus,
- the following additional bonus points are not affected by doubles or redoubles:
  - a small slam, not vulnerable, receives a 500 point bonus,
  - a small slam, vulnerable, receives a 750 point bonus,
  - a grand slam, not vulnerable, receives a 1000 point bonus, and
  - a grand slam, vulnerable, receives a 1500 point bonus.

Write a function called `full-points` that consumes an integer (`level`), a string (`suit`), and three Boolean values (`vulnerable`, `doubled`, and `redoubled`), and produces the full points for the contract. For example:
- `(full-points 3 "NT" false true false)` produces 250 (basic points times 2, plus 50 bonus),
- `(full-points 6 "spades" true false false)` produces 930 (basic points plus 750 bonus), and
- `(full-points 7 "diamonds" false true true)` produces 1660 (basic points times 4, plus 100 bonus, plus 1000 bonus).

You may use the function `basic-points` from part 1a as a helper function for this question.

2. For this question you will submit the functions described in parts 2a, 2b and 2c in a single file.

A **CAPTCHA** (Completely Automated Public Turing test to tell Computers and Humans

Apart) is a test used by websites to try to ensure that a human is accessing the site rather than a bot trying to defeat security. One CAPTCHA technique used by Word Press is to ask people to fill in the blank of a mathematical expression, where some of the values are numbers and some of the values are the words representing the numbers. For example you might see the following:

four + _____ = 11

and you would be expected to fill in the blank with the number 7.
The operation in the mathematical expression could be addition, subtraction, multiplication, or division. There will only be one blank spot, which could appear before or after the operation or after the equals sign. The other two spots could be either words or numbers. All of the values in the calculation will be integers between 1 and 99 inclusive.

(a) Write a function called `word->number` that consumes a string (`word`) representing an integer between 1 and 99. The `word` will be composed of lowercase letters and possibly a hyphen. The function should produce the integer that is equal to `word`. For example,
   - `(word->number "seven")` produces 7, and
   - `(word->number "twenty-four")` produces 24.

   As long as your function produces the correct results for all possible strings, you will receive full marks for correctness. However, for full marks in the Solution Techniques category (which will appear on your marked assignment), your solution should **not** just check the 99 possible words and convert each one into a number. Include helper functions to reduce the total number of question/answer pairs in your solution. If you are having difficulty figuring out how to do this, then finish the other parts of the assignment first, and then come back to this question.
   Hint: If the `word` has a hyphen, it must appear at position 5, 6 or 7 in the string `word`.

(b) Write a function called `fill-in-the-blank` that has four parameters: `arg1`, `op`, `arg2`, and `answer`. The values of `arg1`, `arg2`, and `answer` will be either an integer between 1 and 99 inclusive or the string `"blank"`. Exactly one of the three will have the value `"blank"`. The `op` will be one of the following strings `"+"`, `"-"`, `"*"` or `"/"`. The function should produce the number needed to replace the `"blank"` that makes the mathematical equation: `arg1 op arg2 = answer` true. The value produced will always be an integer between 1 and 99 inclusive. For example:
   - `(fill-in-the-blank 8 "+" "blank" 10)` produces 2,
   - `(fill-in-the-blank 21 "/" 3 "blank")` produces 7, and
   - `(fill-in-the-blank "blank" "*" 5 20)` produces 4.

(c) Write a function called `captcha` that has four parameters: `arg1`, `op`, `arg2` and `answer`. The function will do the same thing as the function `fill-in-the-blank`, except the values of `arg1`, `arg2` and `answer` could be either an integer between 1 and 99 inclusive, a string representing an integer between 1 and 99 inclusive or the string `"blank"`. For example:
   - `(captcha 8 "+" "blank" "ten")` produces 2,
   - `(captcha "twenty-one" "/" 3 "blank")` produces 7, and
   - `(captcha "blank" "*" "five" "twenty")` produces 4.

You may use `word->number` and `fill-in-the-blank` as helper functions for `captcha`.

3. For this question, you will perform step–by–step evaluations of DrRacket 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 webpage:

   `https://www.student.cs.uwaterloo.ca/~cs115/stepping`

   Notes:
   - You will need to authenticate yourself using your Quest/WatIAm ID and password.
   - Once you are logged in, try the "Warmup questions" under "CS 115 Assignment 3", in order to get used to the system.
   - 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.
   - 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.