CS 115 – Spring 2017

Assignment 02

Due: Wednesday, May 24 at 10:00 a.m.

- Do **not** use any conditional expressions (`cond`) on this assignment.
- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch, including helper functions.
- **For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe. Read the Style Guide carefully to ensure that you are following the proper conventions.** ***In addition, your solution must include the definition of constants and helper functions where appropriate.***
- Unless otherwise indicated in the question you may use only 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 at https://www.student.cs.uwaterloo.ca/~cs115/built_in
- 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.
- Read each question carefully for restrictions.
- Test data for all questions will always meet the stated assumptions for consumed values.
- 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.
- 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 any other course staff. Course staff will not accept it as an assignment submission. Course staff will not debug code emailed to them.
- You may post general assignment questions using the discussion groups on Waterloo LEARN. Choose Connect → Discussions. Read the guidelines for posting questions. Do NOT post any code as part of your questions.
- 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.
- 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 a02qY.rkt, where Y is a value from 1 to 4.

Language level: Beginning Student
Coverage: Modules 1 and 2

- Refer to the String documentation found at: https://www.student.cs.uwaterloo.ca/~cs115/resources/strings.pdf for helpful Racket functions.

CS 115 – Spring 2017
Assignment 02
Due: Wednesday, May 24 at 10:00 a.m.

1. A grocery store offers 10% discount on fruits, 25% discount on pastries, and 5% discount on cereal. They also offer 2% discount on other products with price >= $10, and 1% discount on the rest of their products.

   Write a function `grocery-cost` that consumes the price of fruits, price of pastries, the price of cereal, the price of other products with price equal or more than $10, and the price of other products with price less than $10, and produces the total cost of all groceries. For example:
   - `(grocery-cost 20 30 20 50 50)` produces `158`
   - `(grocery-cost 10 15 15 100 30)` produces `162.2`

2. Write a Racket function `gen-username` that consumes two non-empty strings first name and last name, and produces a username from the two strings by concatenating the strings as follows:
   a) the first letter in the first name,
   b) a generated id that concatenates the length of the first name with the length of the last name, and
   c) the last name.

   A username is restricted to a maximum of 10 letters. As a result, only a portion of the last name is included sometimes. You may assume that the first name and last name are provided in lower case. For example:
   - `(gen-username "olaide" "stephen")` produces `"o67stephen"`
   - `(gen-username "kiwi" "alexandria")` produces `"k410alexan"`

   Note: Built-in functions you may find helpful (you do not necessarily have to use them):
   - number->string, string-append, substring, string-length, min, and max.

3. To calculate income tax, assume that the income is over $70,000 and it is taxed according to the following schedule:
   - Basic tax rate: The portion of income up to and including the first $20,000 is federally taxed at 12%.
   - High tax rate: The portion of income between $20,000 and $70,000, is federally taxed at 18%.
   - Additional tax rate: The income earned beyond $70,000 is federally taxed at 22%.
   - There is no provincial income tax for income up to and including $40,000.
   - There is a flat 5% provincial tax for all income in excess of $40,000.

   Write a Racket function called `income-tax` that consumes income, a natural number greater than 70000, and produces the income tax. For example:
   - `(income-tax 110000)` produces `23700`
   - `(income-tax 120000)` produces `26400`

4. The Canadian government issues social insurance numbers (SINs), which uniquely identify their owners. There are specialized algorithms to generate the numbers, but there is a relatively simple algorithm to determine if a given 9-digit number (called the candidate) could be a valid SIN. The algorithm, called Luhn's algorithm, is used to generate a one-digit number from a candidate. If that number is zero, then the candidate is valid. Otherwise, it is not.

   Luhn's algorithm is as follows, for a 9-digit number, $d_1d_2d_3d_4d_5d_6d_7d_8d_9$

   a) Multiply each of d_2, d_4, d_6, d_8 by 2, to get 4 new numbers, p_2, p_4, p_6, p_8
   b) Add together the digits of p_2, p_4, p_6, p_8
   c) Add the digits d_1, d_3, d_5, d_7, d_9 to the sum in the previous step
   d) Determine the remainder when this sum is divided by 10

   For example, consider the candidate 503412896:

   a) 2*0=0, 2*4=8, 2*2=4, 2*9=18
   b) 0+8+4+1+8 = 21 (note: digits of p_2, p_4, p_6, p_8 are added, not the numbers themselves)
   c) 21+5+3+1+8+6 = 44
   d) the remainder when 44 is divided by 10 is 4.

   Write a Racket function `luhns-digit` that consumes a 9-digit candidate (a natural number between 100000000 and 999999999, inclusive), and produces the value from the fourth step (d) in Luhn's algorithm above. For example,

   - `(luhns-digit 503412896)` produces `4`
   - `(luhns-digit 246454284)` produces `0`

   You may find the type conversion operations `string->number` and `number->string` useful. As well, the function `sum-2-digits` is included in the interface file. Read the documentation to determine how it might be useful in your solution.