CS 115 Assignment 4
Due on Wednesday, February 10, 10:00 AM

- You must provide the data definition and template in your solutions **only** when the question specifically indicates they are required for compound data types described in the question. If you create any additional data types that are beyond the question description, your program file should include a data definition and a template for each additional data type. Make sure to comment any templates that you include.
- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch, including helper functions.
- 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 tutors. Course staff will **not** accept it as an assignment submission. Course staff will **not** debug code emailed to them.
- 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 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.
- 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 `a04qY.rkt`, where `Y` is a value from 1 to 3.
- 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 4

Useful structure and data definitions:

```
(define-struct workterm
  (employer city jobtitle weeklysalary weeks))
;; A Workterm is a
;; (make-workterm Str Str Str Nat Nat Nat)
;; Requires:
;;    employer is in all lower case
;;    city is in all lower case
;;    jobtitle is in all lower case


(define-struct app
  (name size OS publisher))
;; An App is a
;; (make-app Str Num Sym Str)
;; Requires:
;;    name is in all lower case
;;    size > 0 is recorded in MB
;;    OS is one of 'ios, 'android, 'blackberry, 'other
;;    publisher is in all lower case


(define-struct smartphone
  (manufacturer OS OSversion favourite))
;; A Smartphone is a
;; (make-smartphone Str Sym Str App)
;; Requires:
;;    manufacturer is in all lower case
;;    OS is one of 'ios, 'android, 'blackberry, 'other
;;    OSversion is composed of
;;    numeric digits 0-9, separated by a single period
;;    favourite is the user's favourite app


(define-struct food
  (name foodgroup unitmass unitprice))
;; A Food is a
;; (make-food Str Sym Nat Num)
;; Requires:
;;    name is in all lower case
;;    foodgroup is one of 'grains, 'vegetables, 'dairy, 'meats
;;    unitmass is recorded in g
;;    unitprice >0 is recorded in dollars per 100 g


(define-struct drink
  (name type container containersize unitprice))
;; A Drink is a
;; (make-drink Str Sym Sym Nat Num)
;; Requires:
;;    name is in all lower case
;;    type is one of 'water, 'juice, 'soft-drink, 'beer, 'wine
```

```
;;     container is one of 'bottle, 'can, 'box
;;     containersize is recorded in mL
;;     unitprice >0 is recorded in dollars per 100 mL


;; A Refreshment is one of:
;;    * a Food or
;;    * a Drink
```

1. Work Term Placements
   Develop a predicate `placement-acceptable?` which consumes a `Workterm` and produces `true` if and only if `jobtitle` ends in `"analyst"` and the total pay for the term is at least $ 6000. Your code is **not** allowed to use a `cond`.

   For example,

   - `(placement-acceptable? (make-workterm "google" "waterloo" "developer" 1000 12))` produces `false`
   - `(placement-acceptable? (make-workterm "blackberry" "waterloo" "ceo" 10000 12))` produces `false`
   - `(placement-acceptable? (make-workterm "sunlife financial" "toronto" "business analyst" 500 16))` produces `true`

2. A Smart Phone

   (a) Write a template for a `Smartphone`.

   (b) Develop a function `initialize-smartphone` which consumes values for the `name`, `OS` and `OSversion` fields of a `Smartphone`, and produces a `Smartphone` where

   - the appropriate fields are populated with the given values, and
   - the string fields of `favourite` are populated with the constant `"none"`, the `OS` field is populated with `'other` and the numeric field of `favourite` is populated with 0.

   For example,

   - `(initialize-smartphone "galaxy" 'android "5.2.1")` produces `(make-smartphone "galaxy" 'android "5.2.1" (make-app "none" 0 'other "none"))`

   (c) Develop a function `update-favourite` which consumes a `Smartphone`, and the field values of `Favourite`, and produces a new `Smartphone` with the smart phone field values unchanged, and the field values of `Favourite` populated with the appropriate parameter values.

   For example,

   - `(update-favourite (make-smartphone "galaxy" 'android "5.2.1" (make-app "candy crush" 2.3 'android "ubisoft")) "angry birds" 1.76 'ios "gameloft")` produces `(make-smartphone "galaxy" 'android "5.2.1" (make-app "angry birds" 1.76 'ios "gameloft")`

3. Food and Drink Costs
   Develop a function `adjusted-price` which consumes a `Refreshment` and returns the same item, with the `unitprice` adjusted according to the following rules. Note, do **not** worry about the number of digits following the decimal point in your computed unit prices.

   - Beer cans are marked up by 5 %.
   - All meats are marked up by 10 %.
   - Juice boxes are marked down by 10 %.
   - Wine bottles of at least 500 mL capacity are marked down by 15 %.
   - All other prices are unchanged.

   For example,

   - `(adjusted-price (make-food "weetabix" 'grains 300 1.50))` produces `(make-food "weetabix" 'grains 300 1.50)`.
   - `(adjusted-price (make-food "t-bone steak" 'meats 1000 15.35))` produces `(make-food "t-bone steak" 'meats 1000 16.885)`.
   - `(adjusted-price (make-drink "frutopia" 'juice 'box 300 0.75))` produces `(make-drink "frutopia" 'juice 'box 300 0.675)`.