CS115 Fall 2012

Assignment 04
Due: 9:00 AM on Wednesday, October 24

Assignment Guidelines:

- 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 copy the purpose directly from the assignment description. The purpose should be written in your own words and include reference to the parameter names of your functions.
- Do not send any code files by email to your instructors or tutors. It will not be accepted by course staff as an assignment submission.
- Read the course Web page for more information on assignment policies and how to organize and submit your work.
- Download the interface file from the course Web page.
- Follow the instructions in the style guide (see the updated version posted on the course web page). Specifically, your solutions should be placed in files a04qY.rkt, where Y is a value from 1 to 4.
- For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions where appropriate.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Read each question carefully for restrictions.
- Use only material from Modules 1 through 5 in your solution. In particular, do not use list abbreviations.

Language level: Beginning Student.
Coverage: Module 5.

1. Create a Scheme function **`made-with-waterloo?`** that consumes **`word`** (a string with one or more lower-case letters). It produces **`true`** if all the letters in **`word`** appear in the word "waterloo", and **`false`** otherwise.

   For example:

```
(made-with-waterloo? "eater") => true
(made-with-waterloo? "toe")   => true
(made-with-waterloo? "alter") => true
(made-with-waterloo? "loop")  => false
(made-with-waterloo? "term")  => false
```

2. The GDP, which stands for Gross Domestic Product, is the market value of goods and services produced within a country in a year. Create a Scheme function **`happy-economy?`** that consumes a list of integers, **`gdplist,`** which represent GDPs of consecutive years. It produces **`true`** if the

integers in the **gdplist** are in an increasing order by at least 5 percent (i.e. the second number is at least 1.05 times higher than the first number, the third number is at least 1.05 times higher than the second number, and so on.), and produces **false** otherwise. The numbers in **gdplist** are all positive integers and are in billion Canadian dollars.

Hint: **gdplist** with zero or one elements produces true for **happy-economy?.**

For example:

```
(happy-economy? (cons 1423 empty))
      => true
(happy-economy? (cons 1500 (cons 1600 (cons 1700 (cons 1900 empty)))))
      => true
(happy-economy? (cons 1500 (cons 1590 (cons 1535 empty))))
      => false
```

3. Write a Scheme function **average** that consumes a list of integers **aloint** and produces the average of the integers in the list if the list is not empty. If the list is empty, **average** produces a symbol **`empty-list**.
   For example:

```
(average empty) => `empty-list
(average (cons 1 (cons 2 (cons 3 empty))))=> 2
(average (cons -5 (cons 12 (cons 0 (cons 3 (cons -2 empty)))))) => 1.6
```

   For testing, if **check-within** is used, an accuracy of two decimal places is sufficient.

4. A list of students are about to move into the newly finished residence building "WaterlooNano". The plan is to move all students with surname initial "A" to "M" before others. Write a Scheme function **moving-order** that consumes a list of student surnames **alonames** and produces a new list that has all the student surnames starting with "A" to "M" followed by all the student surnames starting with "N" to "Z". Note that in the produced list, the names starting with "A" to "M" remain the same order as in the original list **alonames**, as do the names starting with "N" to "Z". All student names start with a capital letter.

   For example:

```
(moving-order (cons "Moore" empty)) => (cons "Moore" empty)
(moving-order (cons "King" (cons "Zheng" (cons "Abbasi"
    (cons "Zorro" (cons "Nehru" (cons "Moore" empty)))))))
     => (cons "King" (cons "Abbasi"(cons "Moore"
          (cons "Zheng" (cons "Zorro" (cons "Nehru" empty)))))))
```