CS 116 Winter 2013

Assignment 01

Due at 10:00am on Wednesday, January 23

- This assignment consists entirely of review material from CS 115 as covered in Module 01. You must use local and abstract list functions (`map, filter, foldr`) where appropriate. All helper functions and constants **must** be defined locally within the main function definition.
- For full marks, each question must use abstract list functions in a non-trivial way. You may NOT use recursion directly in any of your solutions or helper functions.
- Do not use `build-list` or `reverse`. As this assignment is testing material from CS115 and the language level is Intermediate Student, do not use `lambda`.

- For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch, including contracts and purposes for local helper functions. For full marks, it is not sufficient to have a correct program.
- 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.
- 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.
- Download the interface file from the course Web page.
- 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. Specifically, your solutions should be placed in files `a01qY.rkt`, where `Y` is a value from 1 to 4.
- Read each question carefully for restrictions. Test data for all questions will always meet the stated assumptions for consumed values.
- Assignments will not be accepted through email. Course staff will not debug code emailed to them.
- Read the assignment carefully before asking any questions on Piazza.

Language level: Intermediate Student.
Coverage: Module 1

1. Write a function `ones` that consumes a list of natural numbers (`nats`) and produces the number of values in the list whose one's digit (i.e. the least significant or rightmost digit) is 1. For example, `(ones (list 82 231 1 22 1000)) => 2`.

Questions 2 and 3 use the structure `mp`, which represents basic information about Members of the Canadian Parliament:

```
(define-struct mp (name riding party victory-margin))
;; An mp is a structure (make-mp n r p vm)
;; n is a nonempty string (name of the member of parliament)
;; r is a nonempty string (name of the riding the mp represents)
;; p is a symbol (name of the party for the mp),
;;   for example 'CON, 'LIB, 'NDP, 'BQ, 'GRN
;; vm is a positive integer (how many more votes the mp had than the
;;   runner-up in their riding, in the most recent election)
```

Assignment 01

Due at 10:00am on Wednesday, January 23

2. Write a function `members-by-party` that consumes a list of `mp` values (`members`) and a symbol (`party`), and produces the list of `mp`s from `members` whose party field matches the value of `party`. The new list should be in the same relative order as `members`.

   For example,

```
(members-by-party
 (list (make-mp "Jack Harris" "St. John's East, NL" 'NDP 22193)
       (make-mp "Peter Braid" "Kitchener-Waterloo, ON" 'CON 2144)
       (make-mp "Ryan Leef" "Yukon" 'CON 132)) 'CON)
=> (list (make-mp "Peter Braid" "Kitchener-Waterloo, ON" 'CON 2144)
         (make-mp "Ryan Leef" "Yukon" 'CON 132)).
```

3. Write a function `narrow-victories` that consumes a nonempty list of mp values (`members`) and produces a list of the names of all `mp`s who had the smallest margin of victory among `members`. If there are no ties for the smallest margin, then the produced list contains only one name. The produced list must be in the same relative order as `members`.

   For example,

```
(narrow-victories
 (list
  (make-mp "Lee Richardson" "Calgary Centre, AB" 'CON 19731)
  (make-mp "Elizabeth May" "Saanich - Gulf Islands, BC" 'GRN 2144)
  (make-mp "Niki Ashton" "Churchill, MB" 'NDP 4983)
  (make-mp "Dominic LeBlanc" "Beausejour, NB" 'LIB 2585)
  (make-mp "Dennis Bevington" "Western Arctic, NT" 'NDP 2144)))
  => (list "Elizabeth May" "Dennis Bevington").
```

Question 4 uses the new type `student`:

```
;; A student is a list of the form (list id cav passed failed),
;; where
;; id is a natural number (the student id number)
;; cav is a number between 0 and 100, inclusive (student's cumulative
;;     average)
;; passed is a non-negative number (number of credits student has passed)
;; failed is a non-negative number (number of credits student has failed)
```

4. Write a function `candidates` that consumes a list of `student` lists and produces a list of id numbers for students in first year whose strong academic performance makes them candidates for a university scholarship. All students who meet the following criteria are candidates:
   - students are in *first year* (i.e. passed + failed <= 6.0 units),
   - passed >= 4.5 units,

- no courses have been failed, and
- cav is at least 90% or is at least 10% higher than the average of all *first year* students in the list.

The produced list must be in the same relative order as the consumed list.

For example,

```
(candidates
   (list (list 9898 54.7 6.5 0) (list 1234 83.7 6 0)
         (list 1140 57.9 5 0.5) (list 2311 68.5 5.5 0.5)
         (list 6708 61.5 3 0.75)))
  => (list 1234)
```