Deadline: Wednesday, March 11, 2020 at 10:00am on MarkUs

Language level: Beginning Student with List Abbreviations

Coverage: Module 01-05

Files to submit: `a06q1.rkt, a06q2.rkt, a06q3.rkt`

Assignment Guidelines

- Solutions for all questions are expected to follow the requirements of the Style Guide (https://www.student.cs.uwaterloo.ca/~cs115/coursenotes1/styleguide.pdf)

  This includes all relevant design recipe elements, proper use of constants, and proper use of helper functions.

- Submission Details:
  - All solutions must be submitted through MarkUs. Solutions will **not** be accepted through email.
  - For all of the questions verify your basic test results using MarkUs to ensure that your files were submitted properly and are readable on MarkUs.

    Note: passing the basic tests does not guarantee that you will pass all our correctness tests!

- 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.

- Restrictions:
  - Unless specifically allowed in the description of the question, you may only use the built-in functions and special forms introduced in the lecture slides in Module 01-05. For details, see https://www.student.cs.uwaterloo.ca/~cs115/#allowed
  - Read each question carefully to see if any additional restrictions apply.
  - Test data for correctness tests will always meet the stated assumptions for consumed values.

- 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.

Plagiarism: The following applies to all assignments in CS115.

All work in CS 115 is to be done individually. The penalty for plagiarism on assignments (first offense) is a mark of 0 on the affected questions and a 5% reduction of the final grade, consistent with School of Computer Science policy. In addition, a letter detailing the offence is sent to the Associate Dean of Undergraduate Studies, meaning that subsequent offences will carry more severe penalties, up to suspension or expulsion.

To avoid inadvertently incurring this penalty, you should discuss assignment issues with other students only in a very broad and high-level fashion. Do not take notes during such discussions, and avoid looking at anyone else's code, on screen or on paper. If you find yourself stuck, contact the ISA or instructor for help, instead of getting the solution from someone else. Do not consult other books, library materials, Internet sources, or solutions (yours or other people's) from other courses or other terms.

Read more course policies at: https://www.student.cs.uwaterloo.ca/~cs115/#policies

1. **Sum them all!**

   Write a function `sum-them-all` that consumes a list `lloi`, a list that may contain numbers and lists of numbers, and produces the sum of all numbers present in all elements of `lloi`.

   For example:

   - `(sum-them-all empty)` ⇒ `0`
   - `(sum-them-all (list 1 2 3))` ⇒ `6`
   - `(sum-them-all (list 1 (list 2) 3 (list 4 5)))` ⇒ `15`

   Place your definitions into the file **`a06q1.rkt`**.

2. **Approximating Cosine**

   The Taylor series expansion for cosine gives the following infinite series for calculating $\cos(x)$ for any number x, where x is given in radians:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

   You can calculate an approximation of $\cos(x)$ for any given x by adding up some number of terms of this series. The first term is 1, second term is $-\frac{x^2}{2!}$, etc.

   In general, the kth term (starting from $k = 0$) should be $\frac{(-1)^k}{(2k)!}x^{2k}$. Note that this term takes the alternating sign into account.

   Complete the Racket function `approx-cos` that consumes `x` and a natural number `n` and produces the sum of the first $n + 1$ terms of the series above to obtain an approximation of $\cos(x)$.

   If $n = 0$, `approx-cos` should always produce 1 regardless of the value of x.

   If $n = 3$, the calculated value should be:

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}$$

   If $n = 5$, the calculated value should be:

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!}$$

   Notes:

   - Most of your test cases should be using `check-within` (except when $n = 0$) and the built-in `cos` function to show that your function is actually "reasonably close" to `cos`. Specifically, although your function will work for any real number x, the approximation given will have a huge error when $|x|$ is "large"[1]. Therefore, in your testing, you should ensure that:

[1] In fact, the error for this approximation for any given x and n will be less than $\frac{|x|^{2n+1}}{(2n+1)!}$.

- The values of x used for testing satisfy $-2\pi \leqslant x \leqslant 2\pi$ (-360 to 360 degrees).
- You should use the following value for n and tolerance used for `check-within`:

Value of n	$n = 0$	$1 \leqslant n < 9$	$n \geqslant 9$
Tolerance	0 (use `check-expect`)	Do not test	0.02

- As mentioned in A2, the racket `cos` function accepts radians.

For example:

- `(check-expect (approx-cos 0 0) 1)`
- `(check-expect (approx-cos (/ pi 2) 0) 1)`
- `(check-within (approx-cos 0 10) 1 0.02)`
- `(check-within (approx-cos pi 10) -1 0.02)`
- `(check-within (approx-cos (* 3 (/ pi 2)) 9) 0 0.02)`
- `(check-within (approx-cos 10 100) (cos 10) 0.02)`

Place your definitions into the file **`a06q2.rkt`**.

3. **Auto Battler Alliances**

The game *Auto Battler* involves placing chess-like pieces called *heroes* onto a chess-like board. Heroes are assigned one or more *alliances*. For simplicity, we will use strings to represent heroes and alliances.

For this question you are required to complete two parts and submit your solution in the same file. You may use any part of the solution from part (a) in your solution for part (b).

The interface file contains data definitions and some sample data that you may use for your own tests and examples. In your implementation, you should not assume that the specific heroes and alliances that are provided are the only ones that will appear in our tests. However, in your tests, you are not required to introduce new heroes or alliances.

(a) Below is the data definition for Hero Association List, already included in your starter file.

```
;; A Hero Association (HAs) is a (list Str (listof Str))
;; where
;; * The Str is the name of the hero,
;; * The (listof Str) is the list of alliances the hero is a member of.
;;   This list of alliances has no duplicate values.

;; A Hero Association List (HAL) is an association list for
;; Hero Associations. That is, HAL is one of:
;; * empty
;; * (cons HAs HAL)
;; where
;;   Heroes in HAL do not have the same name.
```

For example:

```
(define sample-heroes
  (list (list "Goblin" (list "dwarf" "beast"))
        (list "Orc" (list "beast" "human"))
        (list "Dragon" (list "animal" "mage"))
        (list "Merlin" (list "mage" "human"))
        (list "Gnome" (list "dwarf" "human"))
        (list "Mr. Goose" (list "animal" "beast" "mage" "human"))))
```

In this example, a hero named `"Goblin"` is in the `"dwarf"` and `"beast"` alliances.

Write a Racket function `alliance-count` which consumes an Hero Association List `heroes-al` and an `alliance`. The function produces number of heroes in `heroes-al` that are in the given `alliance`.

For example:

- `(alliance-count empty "?")` ⇒ `0`
- `(alliance-count sample-heroes "beast")` ⇒ `3`

(b) Alliances may have different *levels* depending on how many heroes of a given alliance are on the board. In Auto Battler, a bonus is given to the player if a level of an alliance has been reached. The number of heroes required a particular alliance level depends on the alliance itself.

Below is the data definition for an Alliance Association List, already included in your starter file.

```
;; An Alliance Association (AAs) is a (list Str (listof Nat))
;; where
;; * The Str is the name of the alliance.
;; * The (listof Nat) denotes the number of heroes needed to reach each
;;   level, starting from level 1. This list must not be empty.

;; An Alliance Association List (AAL) is an association list for
;; Alliance Associations. That is, AAL is one of:
;; * empty
;; * (cons AAs AAL)
;; where
;;   Alliances in AAL do not have the same name.
```

For example:

```
(define sample-alliances
  (list (list "dwarf" (list 1 2))
        (list "beast" (list 3))
        (list "human" (list 2 3 4))
        (list "mage" (list 1 3))
        (list "animal" (list 2))))
```

In this example, the `"mage"` alliance has two levels:

- Level 1 is reached if there is one hero on the board who is part of `"mage"` alliance, and
- Level 2 is reached if there are three heroes on the board who are part of `"mage"` alliance.

The maximum level for the `"mage"` alliance is 2. If there are two heroes who are part of the `"mage"` alliance, the level achieved for the `"mage"` alliance is 1. Similarly, the `"human"` alliance has three levels; each level is reached when there are two, three or four heroes of `"human"` alliance on the board, respectively.

Write a Racket function `heroes->levels` which consumes a Hero Association List `heroes-al` and an Alliance Association List `level-reqs`. `heroes->levels` produces the levels of alliances achieved by the heroes in `heroes-al` as defined in `level-reqs` in the form of a list of `(list Str Nat)`, where the `Str` is the alliance name and the `Nat` is the level achieved.

Some further clarifications:

- The orders of the alliances in the list you produce must follow the order that the alliances appear in the Alliance Association List.
- If an alliance has not reached level 1, it should not appear in the produced list.
- You should also assume that all alliances that appear in `heroes-al` will appear in `level-reqs`.

For example, using `sample-heroes` and `sample-alliances` defined earlier:

- `(heroes->levels sample-heroes sample-alliances)` ⇒

```
(list (list "dwarf" 2)
      (list "beast" 1)
      (list "human" 3)
      (list "mage" 2)
      (list "animal" 1))
```

Another example:

-
```
(define my-heroes
  (list (list "Merlin" (list "mage" "human"))
        (list "Gnome" (list "dwarf" "human"))
        (list "Mr. Goose" (list "animal" "beast" "mage" "human"))))
```

```
(heroes->levels my-heroes sample-alliances)  ⇒
```

```
(list (list "dwarf" 1)
      (list "human" 2)
      (list "mage" 1))
```

Place your definitions into the file **`a06q3.rkt`**.