Lab 09: Abstract list functions

Create a separate file for each question. Keep them in your "Labs" folder, with the name **`l`**`ij`**`q`**`k` for **Lab** `ij`, **Question** `k`.

Download the headers for each function from the file `l09-interface.rkt`.

After you have completed a question (except class exercises), including creating tests for it, you can obtain feedback by submitting it and requesting a public test. Follow the instructions given in the Style Guide.

This lab makes use of the following data definitions:

```
;; An Pet is a (list Sym Str Nat), where
;; * the first value is the type of animal
;; * the second value is the associated value.
;; * the third value is its age in years

;; An Point is a (list Num Num), where
;; * the first value is the x-coordinate
;; * the second value is the y-coordinate
```

Language level: Intermediate Student

Important note: Do not write **any** explicitly recursive code in your solutions. Each solution should use at least one of the following abstract list functions: `map`, `filter`, `foldr`, `build-list`, `sort`, `andmap`, `ormap`. You may also find `range` useful.

1. *[Class exercise with lab instructor assistance]*

Create a function *count-even-strings* that consumes a list of strings, *los*, and produces the number of strings in the list that have an even length. This was l05q2.

For example:

```
(count-even-strings (list "a" "ab" "abc" "" "!?")) => 3
```

2. Create a function *switch-case* that consumes a string, *str*, and produces the result of changing the case on all alphabetical characters in *str*, and preserving all other characters. This was l06q2.

   For example:

   ```
   (switch-case "sPoNgEbObIfY") => "SpOnGeBoBiFy"
   ```

3. Create a function *longest-string-length* that consumes a list of strings, *los*, and produces the length of the longest string in *los*. You may assume the longest string in the empty list has length 0. This was l05q3.

   For example:

   ```
   (longest-string-length (list "hello" "there") => 5
   ```

4. Create the function *any-senior?* that consumes *buddies*, a list of Pets, and produces `true` if any pets in *buddies* has age at least 10, and `false` otherwise.

   For example:

```
(any-senior? (list (list 'cat "Mimi" 1)
                   (list 'cat "Duck" 1)
                   (list 'dog "Taz" 16)))
=> true
```

5. Use sort to complete a function *sort-points* that consumes a list of Points, *pts*, and produces a list of the same Point values sorted in increasing order by the sum of the coordinates. Ties between two (or more) Points with the same coordinate sum are broken by ordering the x-coordinate into increasing order.

   For example:

```
(sort-points (list (list 2 1) (list 0 0)
                   (list 0 3) (list -2 4))
=> (list (list 0 0) (list -2 4) (list 0 3) (list 2 1))
```

Optional open-ended questions

Try implementing previous list questions from labs, assignments, and lectures using abstract list functions; like questions 1, 2, and 3 of this lab. Note how `map` can be used with two lists—this will allow you to do lockstep two list problems using `map`.

Helpful tips

Explicit to abstract list functions

Sometimes, when learning abstract list functions, you can get totally lost. A tip to help your learning process is to first write a solution to your problem using explicit recursion. Once you've done that, try translating the solution to abstract list functions.

Racket documentation on ALFs

The Racket documentation on abstract list functions is very useful for finding out how abstract list functions work.