Lab 05: Lists

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 lab-interface05`.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.

Language level: Beginning Student

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

Create the function *add-prefix.* This function consumes a list of strings, *los*, and a string, *pre*, and produces a list of strings formed from the elements of los with the prefix *pre* added in front of each string.

For example:

```
(add-prefix (cons "Dinner" (cons "Lawrence" empty)) “Kraft ”)
=> (cons "Kraft Dinner" (cons "Kraft Lawrence" empty))
```

2. 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.

For example:

```
(count-even-strings
 (cons "cs" (cons "115" (cons "" empty))))) => 2
```

3. Create a function *longest-string-length* that consumes a list of strings, *los*, and produces the length of the longest string in *los*. The longest string in the empty list has length 0.

For example:

```
(longest-string-length (cons “Hello” (cons “World!” (cons “”
(cons “Pasloe” empty)))))
=> 6
```

4. Create a function *or-with-lists?* that consumes a list of Booleans, *lob*, and produces true when at least one element of the list is true. This is essentially recreating the built-in special form or.

For example:

```
(or-with-lists? (cons false (cons true empty))) => true
```

5. Create a function *strings-with-prefix* that consumes a list of strings, *los*, and a prefix, *pre*, and produces a list of all the strings in *los* that begin with *pre*.

   For example:

   ```
   (strings-with-prefix (cons "Wise Wolf" (cons "Man" (cons "wise
   Donkey" (cons "" empty)))) "Wise ")
   => (cons "Wise Wolf" empty)
   ```

6. Create a function next-list that consumes a list, *alist*, and an item, *item*, and produces either the element in *alist* that appears after the **first** occurrence of *item* or the string "`none`" if item is either the last element in *alist* or not in *alist*.

   For example:

   ```
   (next-list (cons 2 (cons "a" (cons -4 (cons 2 empty)))) 2)
   => "a"
   ```

Optional open-ended questions

Create a function that consumes a list of Boolean values, representing the binary encoding of a number (true = 1 and false = 0), and produces the binary encoding of a number one greater. Consider functions that double a binary number or make a binary number one smaller.

If you used the built-in special form or for question 4, try to recreate your function without using it. If you didn't use or, try to recreate your function using or.

Helpful tips

String documentation

The string documentation has helpful information on strings and characters. You may find it under the heading Resources on the course website.

List templates

Beginning from the list template ensures that you will use structural recursion. It also makes the thought process of designing functions easier!