Lab 08: Processing two 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 `l08-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 association (As) is (list Num Str), where
;; * the first item is the key,
;; * the second item is the associated value.

;; An association list (AL) is one of
;; * empty
;; * (cons As AL)

;; Note: All keys must be distinct.
```

Language level: Beginning Student with List Abbreviations

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

Create a function *extract-values* that consumes a list of keys, *keys*, and an association list (as defined in Module 5), *assolist*, and produces the list of values associated with the keys in *keys*. The order of the produced list matches the order the values appeared in the association list.

For example:

```
(extract-values (list 5 2)
                (list (list 1 "a") (list 4 "b")
                      (list 2 "c") (list 3 "d")
                      (list 5 "e")))
=> (list "c" "e")
```

2. Create function *extract-bad* that consumes two strings, *bad-part* and *whole-string*, and produces the string formed by removing each letter in *bad-part* from *whole-string*. The characters in *bad-part* will appear in *whole-string*, in order, though not necessarily consecutively.

   For example:

   ```
   (extract-bad "aadba" "abracadabra" => "brcaar"
   ```

3. Create a function *subseq-string* that consumes two strings of letters in lowercase (*pattern* and *target*), and produces a string showing how much of *pattern* can be found in *target*. The string that is produced will have the same letters as *target*, but with some changed to upper to show where and how much of *pattern* can be found. *pattern* will be found at most once in *target*, and you may assume the first matching character encountered should be matched. You may want to try writing a helper that handles lists of characters.

   For example:

```
(subseq-string "hat" "chordality") => "cHordAliTy"
(subseq-string "hat" "hand") => "HAnd"
(subseq-string "hat" "hhat") => "HhAT"
(subseq-string "hat" "hathat") => "HAThat"
(subseq-string "atebr" "zebrazebra") => "zebrAzebra"
```

4. Create a function *clicker-grades*, that consumes two lists of symbols of equal length. Each symbol will be one of ‘a, ‘b, ‘c, ‘d, or ‘e. The first list, *corr-answers,* contains the correct answers to the clicker questions. The second list, *stu-answers*, contains a student's answers to the clicker questions. You may assume that the first element of *corr-answers* corresponds to the first element of *stu-answers*, and so on. *clicker-grades* produces the number of correct responses the student gave.

   For example:

```
(clicker-grades (list ‘a ‘a ‘b ‘c) (list ‘a ‘d ‘b ‘d)) => 2
```

5. Create a function *clicker-grades-absent*, similar to question 4. However, for this question, the student may have been absent for some clickers in the most recent lecture. Thus, *responses* may be shorter than *corr-answers*. Note that *responses* cannot be longer than *corr-answers*. It is still true that the first element of *corr-answers* matches the first element of *responses*, and so on. *clicker-grades-absent* still produces the number of correct responses the student gave.

   For example:

```
(clicker-grades-absent (list ‘a ‘a ‘b ‘c) (list ‘a ‘b)) => 1
(clicker-grades-absent (list ‘d ‘e ‘a) (list ‘d ‘e ‘b)) => 2
```

Optional open-ended questions

Same template, different function?

In question 1, we specified that the items in the final answer should appear in the same order that they appeared in the association list. What if we wanted the items to appear in the order that we ask for their keys? Try to write an alternate version of extract-values that does this.

A calculator?

Create a function list-calculator that consumes two lists, operators and operands, where the length of operands is exactly two times that of operators. Operators may contain only the following elements: '+, '*, '/, '-, representing addition, multiplication, division, and subtraction, respectively. Operands is a list of numbers. list-calculator will apply the operator from operators to the two operands from operands, the first of operators goes with the first and second of operands, and so on. This will create a list of the same length of operators, with the results of each calculation.

Calculating a CS 115 final grade

Your final grade in CS 115 is calculated as the sum of weighted grades. The assignments are 20% of this final grade, each assignment is then worth 2%. If one were to encode their assignment grades as a Racket list, it might look like (list 0.87 0.85 0.95 0.60) for the first four assignments. This means the student received 87% on a01, 85% on a02, etc. If one were to encode those grades after they've been weighted it would be (list 0.0174 0.017 0.019 0.012). Try creating a function that consumes a list of evaluations (a list containing a label for the evaluation, the mark you got, and the weighting) and a list of weighted percentages representing the marks you've already received/calculated (it could be empty). The function uses these to calculate a final grade.

Helpful tips

Two lists templates

In CS 115, we have split two list recursion into three separate cases. When dealing with any two list question, your first idea should be which of the cases it is. Once you have figured that out, you can begin from one of the templates.

Case 1 is taking a list "along for the ride". This is for when one list doesn't change *at all*.

Case 2 is processing lists at the same rate—"in lockstep". This is for when the lists have the same length and items in a given position correspond to each other.

Case 3 is for processing lists at a different rate, the most complex. This case presents many possibilities, and applies when the other two don't.