COMP 50 Lab: Two Lists at Once

October 9–10, 2013

This lab gives you supervised practice working with functions that consume two lists (and possibly other arguments). During lab, you will have time to complete only one of the two problems:

- The first problem gives you practice at processing two lists in lockstep, and it also gives you practice working with multiple lists by "sealing" all lists except one.
- The second problem gives you practice at handling two lists in the general case.

If you choose to complete both problems, you will have had practice working with all three kinds of functions that operate on multiple lists. And with either problem, you will also have practice using the BSL functions `explode` and `implode`, which convert between strings and lists.

Every function should include enough test cases for all shapes of input. So for example, if a function consumes two lists, it must have *at least* four test cases.

The problems

1. *Symmetric substitution cipher.*
   A substitution cipher (weakly) hides a message by substituting one letter for another. The cipher is symmetric if doing the same substitution twice reproduces the original message. Such ciphers are easy to break, but if all you want to do is keep people from stumbling over information accidentally, they are useful.

   Here is a diagram of a popular substitution cipher:

   ```
   Decryption Key
   A|B|C|D|E|F|G|H|I|J|K|L|M
  -------------------------
   N|O|P|Q|R|S|T|U|V|W|X|Y|Z
   (letter above equals below, and vice versa)
   ```

   Define a function `substitute-char` that takes as arguments a 1-character string `c` and two lists of 1-character strings `top` and `bottom`. The lists `top` and `bottom` together may be assumed to contain no duplicates.

   - If the string `c` appears in the list `top`, the function should return the corresponding string from the list `bottom`.
   - If the string `c` appears in the list `bottom`, the function should return the corresponding string from the list `top`.
   - If the string `c` appears in neither `top` nor `bottom`, the function should return `c`.

   Here are some examples:

```
(check-expect (substitute-char "H" (explode "ABCDEFGHIJKLMabcdefghijklm")
                                   (explode "NOPQRSTUVWXYZnopqrstuvwxyz"))
              "U")
(check-expect (substitute-char "q" (explode "ABCDEFGHIJKLMabcdefghijklm")
                                   (explode "NOPQRSTUVWXYZnopqrstuvwxyz"))
              "d")
```

Lists are all very well, but for people it is easier to work with strings. To finish this problem, *define a function* `two-row-cipher` that takes three arguments: a *cleartext* string, a top-row string, and a bottom-row string. The function converts the cleartext to a corresponding *ciphertext* by applying the substitution to each character; it returns the ciphertext, also as a string. I recommend that you use `substitute-char` to substitute for each character of the cleartext string, and that you use BSL functions `explode` and `implode`.

Example:

```
(check-expect
  (two-row-cipher "Fishy" "ABCDEFGHIJKLMabcdefghijklm"
                          "NOPQRSTUVWXYZnopqrstuvwxyz")
  "Svful")
```

Use your function to decrypt the following message, which contains an observation made by a keen student of human nature:

> Vg vf n gehgu havirefnyyl npxabjyrqtrq, gung n fvatyr zna va cbffrfvba bs n tbbq sbeghar zhfg or va jnag bs n jvsr.

2. *Anagrams*.
   Two words are anagrams if they are made from exactly the same letters:

   - `"fowl"` and `"wolf"` are anagrams
   - `"fowl"` and `"foul"` are not anagrams
   - `"read"` and `"dear"` are anagrams
   - `"read"` and `"dread"` are not anagrams
   - `"read"` and `"red"` are not anagrams
   - `"stropped"` and `"postered"` are *not* anagrams

   Define a function that takes as arguments two strings and returns a Boolean saying if they are anagrams. Use the string-conversion function `explode` from the BSL.

   Follow the design recipe. When you reach the stage of filling in the code template, *develop a wish list*, and **show your wish list to the lab staff**.

Submitting the lab

Ten minutes before the end of the lab, put the following text at the beginning of a DrRacket file. You may use an empty file or the source code you have been working on:

```
#|
What I did during this lab:
   (you fill in this part)

What I learned during this lab:
   (you fill in this part)

|#
```

The lab staff will help you articulate what you learned.

Finally, *submit this file through the handin server* as `lab-lists`. You will need to submit it using *two* usernames connected by a + sign, as in

```
Jane.Doe+Richard.Roe
```

You submit using Jane Doe's password.