## CS 2500, Spring 2013 Problem Set 6

Due date: Tuesday, February 19 @ 11:59pm

Programming Language: Beginning Student Language with List Abbreviations

## What to submit:

You must work on this problem set in pairs. Homework partners have been chosen randomly and posted on Blackboard. You must work on and submit the homework with your partner. Using Blackboard, submit a single Racket file containing all of the code and documentation for this assignment. Place your names and husky email addresses in a comment at the beginning of your file.

Name your file: hw6-lastname1-lastname2.rkt

This problem set continues the study of self-referential unions. Some of the problems cover functions that process two arguments from classes with complex (self-referential) data definitions. You must follow the design recipe in your solutions: graders will look for data definitions, contracts, purpose statements, examples/tests, and properly organized function definitions. For the latter, you must follow templates. You do not need to include the templates with your homework, however, unless the question asks for it.

## Problem 1.

In this problem, you will implement functions for a simple version of a social networking system such as Facebook.

- a) A profile consists of the user's name, location and relationship status and a lof (list of friends). A friend consists of a name, location and relationship status. Write data definitions and provide examples of data for profile, friend, and lof.
- b) Write the templates for profile, friend and lof.
- c) Write a function, total-friends, that consumes a profile and produces the total number of friends that the user has.
- d) Write a function add-friend that consumes a profile and the friend to add, and returns a profile. If the friend is not in the lof of the profile, then the friend is added to the lof. Otherwise, the profile is returned unchanged.
- e) Write a function un-friend that consumes a profile and the friend to delete, and returns a profile. If the friend is in the lof of the profile, then the friend is deleted from the lof. Otherwise, the profile is returned unchanged.
- f) Write a function friends? that consumes a profile and a profile and produces a Boolean. The function returns true if the user of the first profile is a friend of the user of the second profile and vice versa, and false otherwise.

g) Write a function print-friends which consumes a profile and produces a string with all of the friends' names.

## Problem 2.

Using Unicode characters, we can get a very close approximation of upside-down text for lowercase letters. The lowercase letters have been encoded into Unicode, creating a full set of upside-down lowercase letters (more or less). (Try typing in "\u0250" in the interactions window in DrRacket.)

Design a program that will consume a string of lower-case letters and produce the string upside-down. Create helper functions as needed.

A few helpful hints to get you started:

```
You can use explode to produce a list of 1-letter strings from a string:

;;explode: String -> ListofString

;;translates a string into a list of 1-letter strings

;;example: (explode "cat") -> (list "c" "a" "t")

And implode will allow you to produce a string from a list of strings:

;;implode: ListofString -> String

;;concatenates a list of 1-letter strings into one string

;;example: (implode (list "c" "a" "t")) -> "cat"
```

We don't want you to labor over the Unicode entry, so a table is provided (you can also download this from Piazza):

```
;; A LOS is one of
;; empty
;; (cons String LOS)
;;
;; A Table is one of
;; empty
;; (cons (cons String (cons String empty)) Table)
;; build-table: LOS LOS -> Table
;; build a table from two lists of strings
;; assume: lists are equally long
(define (build-table domain range)
         (cond
                  [(empty? domain) '()]
                  [else (cons (list (first domain) (first range))
                                                                       (build-table (rest domain) (rest range)))]))
(check-expect (build-table (list "a") (list "b"))
                                                              (list (list "a" "b")))
(define table-1
         (build-table
             (explode "abcdefqhijklmnopqrstuvwxyz?!'")
             (explode "eqcpg" shanfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfardboumsfard
```