CS 2500, Spring 2012
Problem Set 6

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

Programming Language: Beginning Student Language with List Abbreviations

For Problem Set 3 and later, homework is submitted via the automated homework server.
Note: Hardcopy submissions not accepted. Email submissions not accepted.

You must work on this problem set in pairs. Homework partners have been chosen randomly and posted on the Piazza discussion board. You must submit the homework with your partner. There will be a penalty for submitting without your partner.

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 design templates, but be sure to comment them out.

Part 1 – HtDP problems:

14.1.3, 14.1.5, 14.2.4

Part 2 –
In this part, you will implement functions for a simple version of a social networking system such as Facebook. Your solutions will be graded partly for their use of helpers and adherence to templates.

1. 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. A `lof` is a `list of friend`.
   Write data definitions and provide examples of data for `profile, friend, and lof`.
2. Write the template(s) for `profile, friend` and `lof`.
3. Write a function `total-friends` that consumes a `profile` and produces the total number of friends that the user has.
4. 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.
5. 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.

6. 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.
7. Write a function `print-friends` which consumes a `profile` and produces a string with all of the friends' names.