CS 127: Data Structures	**Winter 2006**
Handout 4: Problem Set #2 (Due: 11:59p, Monday, 23 January 2006)[1]	*18 January 2006*

Those who do not learn from history are doomed to repeat it.
— George Santayana (1863–1952).

This assignment is to be done in your assigned groups. Two things to remember in your groups:

- If there are n people in your group and you work for t hours on the assignment, you should aim to be "driving" (i.e., typing) approximately t/n hours each.
- Ensure that all members of your group are at a similar spot in terms of comprehension. If you're more familiar with a particular concept or with Java syntax or something, try to let your partner do more typing. If you're less familiar, be a little more aggressive about taking the reins. (And don't be embarrassed! You're here to learn.) This strategy will help both you and your partner learn, and it will also help to make sure that your group catches more mistakes, because both of you will mentally ready to catch bugs. Also, I will be assigning homework grades based in part on your peer evaluations, and I will notice if there is a term-long (= long-term?) trend of you doing too large or too small a share of the work.

This assignment asks you to add some functionality to a bare-bones web browser that is provided to you. We will start with some very basic code that renders a web page in a window and that can read and interpret clicks on hyperlinks. In this week's assignment, you will add some functionality to the web browser's GUI (graphical user interface) and implement the "history" functionality in the web browser. (In this context, "history" refers to a list of the last web pages visited, usually in some kind of time order.)

As always, if there are any questions that come up, please ask away! David, George, and the lab assistants are all great places to go for help.

0. Estimate the amount of time you spent on this problem set, and write it at the top of your `ps2.txt`.

1. So far you've seen two basic data structures in class—the array and the linked list. Perhaps unsurprisingly, we can use either one to maintain the history of pages viewed in the browser. In this assignment, you'll implement the data structure using a linked list. We will want to be able to display the 10 most-recently viewed pages sorted in two orderings: (1) from most-recently viewed to least-recently viewed, and (2) from least-recently viewed to most-recently viewed.

   In this question, you'll build a class that will do the work required to implement the linked-list option. Because we want to be able to report histories reading in both directions, you'll need a doubly linked list. Write a class named `DoublyLinkedList` that implements a doubly linked list in Java. Make sure that your class is generic (i.e., that it uses generics). You must implement this class from scratch; you may not use Java's built-in `LinkedList` class for this assignment. You are welcome, however, to use the book or your notes from class on singly linked lists (and the code) as a guide to get you started.

   There are some standard—and non-standard—linked-list methods that you'll need "for sure" to do the next part of this assignment, and there are some methods that you almost certainly won't use. You don't have to write the methods that are unneeded for the next question, so you need to sit down and think about what methods you'll need in your linked-list class. Read over the entire problem set, decide what you'll need in your linked-list class, and prepare a design document for your linked-list class. (See the course web page.) In general you'll have to make more design decisions than those required for this assignment; here you need only describe the design of your linked-list class.

   Email David your design document (in plain text!) by 7:00p on Friday, 20 January 2006.

[1] Design document due by email to David by 7:00p on Friday, 20 January 2006. See question #1.

2. Go to the course web page and download `Minibrowse.java`. Compile and run it. Play around with it a bit to learn how it works. Try removing lines, adding lines, etc., and see how it affects the behavior of the program.

   Now we will add the "history" functionality to the browser. First, modify the Minibrowse code so that it maintains a list of the last 10 web pages visited.

   - If a user visits the same web page more than once, include in the list more than once. In other words, duplicates are fine.
   - The list must contain at most 10 web pages at a time. So, if a user visits more than 10 web pages, every time the user visits a new web page you must remove the oldest page in the list. It is up to you to decide how to remove the oldest page and insert the newest one. In your `ps2.txt` file, briefly discuss your methodology and why you chose it.

   Next, add two buttons to the browser's toolbar: a "History+" and a "History−" button. When pressed, each button should open a popup menu that lists the last 10 web pages visited. The "History+" button should display the pages from newest to oldest, while the "History−" button should display the pages from oldest to newest. You should do this solely by traversing your list; sorting is not necessary.

   The file `OutputWindow.java`, also on the course web page, will provide some examples of how you might display the history window.

3. Describe in a few paragraphs (I recommend including pseudocode) how you could implement the history using an array instead of a linked list. You might want to take a look at the array-based implementations of stacks or queues that are in the book if you're having trouble. In your `ps2.txt` file, discuss why you might prefer to use the array-based implementation or the list-based implementation. Argue why one of them is better, or argue why "it depends" (and say on what it depends).

4. *Bonus—utterly unrequired.* If you have finished everything up to this point and you have time and inclination left (and your code for the previous questions works brilliantly and is well documented and clear), then implement the `amihotornot` deletion feature. (What if `http://www.amihotornot.com` is in your history and you'd really like to delete it before someone you're trying to impress walks in the door?) Add a "Clear last" button to the browser that, when clicked, removes the most-recently viewed page from the history. Also describe in `ps2.txt` the most efficient implementation you can manage for the `amihotornot` feature in the array case.