CS 127: Data Structures **Winter 2006**
Handout 5: Problem Set #3 (Due: 11:59p, Monday, 30 January 2006) *25 January 2006*

I take two steps forward,
I take two steps back.
— Paula Abdul (b. 1962), "Opposites Attract."

Bartender: *Uh-oh. You're not trying to lose weight on one of them fasts, are ya?*
Palindrome: *Doc, note, I dissent. A fast never prevents a fatness. I diet on cod.*
— Riders in the Sky, "Ballad of Palindrome."[1]

This assignment is to be done *individually.* As always, you are welcome to talk to David, George, and the lab assistants for help. Again as always, you're also welcome to give "consulting" help to each other. (Consulting help includes talking about strategies for solving problems or looking over another student's shoulder and pointing out a bug. It *never* includes giving physical or electronic possession of your code to another student.) Cite your sources, including any of the above.

This assignment has a few goals: to give you experience with stacks and recursion, to give you a chance to explore their uses, and to continue with our project of building a web browser by adding Back and Forward buttons to MiniBrowse. Note that most of the questions below are independent of each other, so you can work on them in whatever order you please.

0. Estimate the amount of time you spent on this problem set, and write it at the top of your `ps3.txt`.

1. In a new directory `ps3`—having a separate directory for this problem set is especially important, so that you don't overwrite your previous PS2 submission—copy `Minibrowse.java` either from your `ps2` directory or from the course web page. Also download `RealStack.java` from the CS127 web page.

   Write a generic class `MyStack⟨E⟩` that implements the interface `RealStack⟨E⟩`. Implement the stack using whatever underlying data structure you please, including the built-in classes `ArrayList` or `LinkedList`, or the `DoublyLinkedList` class that you wrote for the last assignment. (The only thing you can't use is Java's built-in `Stack`.) Also write a `toString()` method; it will serve you in good stead when you work on later parts of this assignment.

2. Start with the `Minibrowse` code that you handed in for PS2. (You can also start from scratch; success in this assignment does not depend on a working PS2.) Modify `Minibrowse` to add "Back" and "Forward" buttons to the browser, and associate a stack with each button. Then modify Minibrowse to match the behavior of "back" and "forward" in standard (non-mini) web browsers, as follows:

   - Whenever a user types in a web page or clicks on a hyperlink, its URL is pushed onto the stack associated with the Back button.
   - If the user clicks on the Back button, the current URL should be pushed onto the stack associated with the Forward button before a URL is popped off the back-button stack and displayed.
   - Likewise, whenever a user clicks the Forward button, the same thing should happen in reverse. Make sure to handle the special case that if a user types in a URL into the address bar, the Forward button should be cleared. (Play around with a "real" browser and see how the Forward button behaves when you typed in a new web page).

   Don't violate the stack's interface! The only way to store URLs for the forward and back buttons is by pushing onto a stack. The only way to access URLs is via `peek()` and `pop()`. In order to clear a

[1] See `http://snipurl.com/lv82`.

stack (such as when you need to clear out the Forward button), you must do so by popping everything off the stack until it is empty.[2] If you wish, you may write a `clear()` method within your Minibrowse code that does multiple pops to make your life easier, but the stack itself should remain purely a stack.

3. A *palindrome* is a string that reads the same backwards as forwards. You will need to strip the whitespace, punctuation, and capitalization from the string before (or while) you test it for palindromicity.[3]

   Write a class called `Palindrome` that includes the following methods:

   - `public static boolean isPalindromeStack(String s)`. This method returns true if the specified string is a palindrome, and false otherwise.

     Your palindrome tester must operate as follows: one by one, push the characters of the string s onto a stack. When you are out of characters, pop the characters off the stack and see whether the resulting string s' (where the ith character of s' is the ith character popped off of the stack) is identical to s.
   - `public static boolean isPalindromeRecursive(String s)`. This method returns true if the specified string is a palindrome, and false otherwise.

     This palindrome tester must operate recursively. Before you start coding, consider this question: how can I break this problem down into a smaller problem of the same type? For example, a string has even length if the string with the first and last letters removed also has even length. When is a string a palindrome? (When is a string that ends with a punctuation mark a palindrome?)
   - A main method to test the above methods.

   You may be interested in page 700 of Koffman & Wolfgang.

In your `ps3.txt`, include a brief description of the design decisions that you made in writing your stack class, anything special about your Minibrowser, and anything else interesting about your implementations.

4. Go to the course website, and use the anonymous feedback form to give me some comments about the class so far. Is the pace too fast? Too slow? Are lectures clear? Are they interesting? What would you change if you could?

   Your answer to this question *will* impact your grade! Your score will the number of responses that I receive and judge to be from distinct sources, divided by the number of people in the class. (If this ratio is greater than one, you will receive zero points.)

[2] Yes, there are other ways to do this operation, including clearing the stack `S` by setting `S = new MyStack()` and letting the garbage collector handle the deletion of the old stack. Don't do it this way.

[3] No, that's not a word. But neither is "prefector," so let's not make too big of a deal of it, eh?