CSc 127B — Introduction to Computer Science II
Fall 2015 (McCann)

http://www.cs.arizona.edu/classes/cs127b/fall15/

Section 9: Queues and a Taste of Linked Lists

Pair up with anyone who is agreeable to pairing up with you, pick the first driver, and let's get to work!

PART I: A `StringBuilder` Queue

In Section 8, you created a stack class that stacked characters using a `StringBuilder` object as the stack's representation. In Part II of this activity, you will have need of that stack class, and of a queue class that also holds characters. In this part you'll create that queue class, which also uses a `StringBuilder` object as its data structure.

1. Open DrJava, and create a class named `QueueSec9` that uses a `StringBuilder` object to hold characters in a queue. Here are the method signatures of the methods that your class needs to provide:
   - `QueueSec9()` — the constructor; creates an empty queue
   - `boolean isEmpty()` — true if no characters in the queue; false otherwise
   - `int getOccupancy()` — quantity of characters in the queue
   - `void enqueue(char)` — add char to the rear of the queue
   - `int dequeue()` — removes the front character, returns -1 if empty
   - `int peek()` — shows the character at the front of the queue, returns -1 if empty

   Make it happen! (Hint: Looking at your `StackSec8` class from Section 8 might help.)
2. Copy and paste the main program file, `QueueSec9Main.txt`, from the class web page into your `QueueSec9` class.
3. Compile and run `QueueSec9`. If necessary, debug your methods until you're getting the output specified in `QueueSec9Main.txt`'s comment.

✔ **CHECKPOINT 1** Raise your hand. Your SL will come over and verify that your class is working.

PART II: Detecting Palindromes with a Stack and a Queue

A palindrome, as your SL explained at the start of section today, is a word or phrase whose letters appear in the same order left-to-right and right-to-left. For example, `I'm a lasagna hog, go hang a salami.` is a palindrome (we ignore spaces and punctuation symbols).

One way to test a string to see if it is a palindrome: Take each letter in the string and both push it onto a stack and enquque it into a queue. Then, repeatedly pop the stack and dequeue from the queue. If the pairs of characters removed do not match, then the string is not a palindrome. But, if you empty both data structures and all of the pairs of characters have matched, the string is a palindrome.

1. Locate your version of `StackSec8` from last week and load it into DrJava. If neither you nor your partner have a working version, you can use the version on the class web page (under the link to this handout, not in the Section 8 area).

(Continued . . .)

2. Using a pen/pencil and paper, use the algorithm described at the top of this part, and drawings of a stack and a queue, to test each of these four small strings:

   (a) `kayak` (a palindrome)

   (b) `abcb` (one 'a' short of being a palindrome)

   (c) `bcba` (same, but on the opposite end)

   (d) `abcdeba` (not a palindrome; 'c' doesn't pair with 'e')

   (The point is to get a feel for how the algorithm works, which should make it easier to write a program that carries out that algorithm.)

3. Write a Java program named `PalPal.java` (for 'Palindrome Pal') that takes a string from the command line and tests it to see if it is a palindrome using the stack/queue algorithm. The following execution examples show the behaviors your SL will expect your program to demonstrate:

```
> java PalPal
Usage example:  java PalPal "Yo, banana boy!"

> java PalPal ""
"" is a palindrome!

> java PalPal "Sit on a potato pan, Otis!"
"Sit on a potato pan, Otis!" is a palindrome!

> java PalPal "abcb"
"abcb" is not a palindrome;
the problem is that 'a' doesn't match 'b'.

> java PalPal "bcba"
"bcba" is not a palindrome;
the problem is that 'b' doesn't match 'a'.

> java PalPal "abcdeba"
"abcdeba" is not a palindrome;
the problem is that 'c' doesn't match 'e'.
```

   Hint: Remember that the `Character` wrapper class has static methods that can test characters to see if they have various properties.

   $\boxed{\checkmark\ \textbf{CHECKPOINT 2}}$ Raise your hand. Your SL will come over and check that your program is working (and is following the given algorithm).

PART III: Tracing Linked List Code

Writing correct linked-list code requires the ability to trace through the code to understand the operations that need to be performed, and the order in which they need to be performed. Your SL did a tracing example earlier. Now it's your chance to try it.

1. Find that pen/pencil, and the paper, you used in Part II; they couldn't have gotten far.

2. Draw yourself three example linked lists: One with three nodes, one with just one node, and one with no nodes (`head` just references `null`). Each node should hold one integer value (and a next field, of course).

(Continued ...)

3. As your SL did at the start of section today, trace the execution of the following linked-list code on each of your three example lists. Do this tracing in a "pair programming" fashion; that is, one person uses the pen and paper to trace the execution, the other is the helper.

   This code is supposed to compute the sum of the values held by the nodes in the list, but it has some logic problems. While you're tracing it on your example lists, figure out why the code doesn't work, and fix it so that it does work on all three of your lists. Be sure to write down the necessary changes; your SL will want to know what they are.

```
private static int sum (LLNode head)
{
    LLNode temp = head;
    int total = head.getData();
    while (temp != null) {
        temp = temp.getNext();
        total += temp.getData();
    }
    return total;
}
```

4. Switch roles; the helper is now the tracer, and the tracer is now the helper. (If you were switching roles between lists above, great; keep doing that!)

5. We aren't going to tell you what the following code does; your job is to figure that out by tracing the execution on your example lists. There aren't any bugs (that we know of!) in this method; all you have to do is figure out what it does.

```
private static LLNode mystery (LLNode r)
{
    LLNode u = r, t = null, n = t;
    while (u != null) {
        t = n;
        n = u;
        u = u.getNext();
        n.setNext(t);
    }
    return n;
}
```

6. Did you enjoy the lack of comments and the meaningless variable names? Take that to heart when writing your own code. Good programming style is important, which is why a significant portion of your grade on programming assignments is devoted to it.

✔ **CHECKPOINT 3** Raise your hand. Your SL will come over and see how you did.

PART IV: Clean Up!

1. Log out of your computer; pick up your papers, writing implements, cell phones, trash, etc.; push in your chair(s).

✔ **CHECKPOINT 4** Raise your hand. Your SL will come over and use his or her tricorder to count the bacteria lingering on your keyboard. (Don't expect him or her to say, "They're dead, Jim" unless your name is actually 'Jim.')