**Iterator Lab**

Iterators and ListIterators are private objects inside each Collection object. After you instantiate an iterator, you use the *iterator's* methods to traverse and modify the list. Notice on the cheat sheet the iterator's methods it.hasNext(), it.next(), it.remove(), it.add(o), it.set(o).

It is common (see the cheat sheet comments on ListIterator's add) to explain the action of an iterator by imagining a current pointer that is between two data items. Suppose a ListIterator has been instantiated so that current is before the first item: ^ **A B C D E**

1. Show the iterator's position after it.next(); it.next(); **A B ^C D E**

2. Then do it.remove(); Write the list. **A B ^D E**

3. Then do it.next(); add(**X**). Write the list. **A B ^X D E**

1. What happens if you try to list.add(0) or list.remove(i) when an iterator is active?

**Concurrent modification exception**

5. A *for-each* loop or an Iterator: how do you choose?

**For-each when you don't want to edit the contents without error.**

**Iterator for when you want to edit the contents without error.**

6. An Iterator or a ListIterator: how do you choose?

**Iterators apply to general lists (sets, lists, maps). ListIterators apply to Lists only. Listiterators also have more methods. ListIterators have previous(), nextIndex(), previousIndex()**

1. Iterating and deleting many elements in a LinkedList is O(n). Explain why.

**You can just loop through the list and remove as necessary.**

1. Iterating and deleting many elements in an ArrayList is O(n2). Explain why.

**Since we're working with many elements, we may have to iterate over the list multiple times. Say we have to iterate over an arraylist [1,2,3,4,5,6,7,8,9,10] and delete 10, 9, 8, etc. The iterator would have to loop through the array each time, each time the size getting smaller, so the running time would be n + n – 1 + n – 2 + n … 0, which essentially equals n^2**

9. Complete the IteratorLab\_shell, using either for-each loops or iterators—not regular for-loops. Notice Lines 11-12, where they made the for-each loop backwards compatible with primitives. The replaceNeg method on Line 17 replaces all negative numbers with 0.