Thursday, February 22

Warmup
What are the instance variables for the sequence with a partially-filled array?
What is the class invariant?

    double[] data
    int manyItems
    int currentIndex

manyItems indicates how many items are in the sequence

data[0] to data[manyItems-1] contain the items in the order the user requested and we don’t care about the other elements in data

currentIndex contains the index of the current item if there is one
if there isn’t one, currentIndex equals manyItems

Linked Lists - Chapter 4 (not on test 1)
Array - use index to access part of a chunk of allocated storage
Linked List - connect the pieces of allocated storage

linked list: a series of items arranged one after the other with each connected to the next by a link

node: contains data and link (usually to next node)
    someType data;
    node link;

null reference: null; value at end of list

intNode.java - in examples directory
    // Invariant of the IntNode class:
    //   1. The node's integer data is in the instance variable data.
    //   2. For the final node of a list, the link part is null.
    //      Otherwise, the link part is a reference to the
    //      next node of the list.
    private int data;
    private IntNode link;

looked at listLength method along with picture
    public static int listLength(IntNode head) {
        IntNode cursor;
        int answer;

        answer = 0;
        for (cursor = head; cursor != null; cursor = cursor.link){
            answer++;
        }

        return answer;
    }
draw a lot of pictures

see picture handout in notes directory - LLreview.pdf
insert after existing node, insert at front
we will discuss remove after existing and remove front on Tuesday

question about declaring tail and getting its value
don't need to have a tail pointer
need a head, but initially it is null

    // sketch of code
    IntNode tail;  // no data or link
    IntNode list = new IntNode(42 null); //create start of list
    ...  // list was built elsewhere
    for (tail = list; tail != null && tail.link != null; tail = tail.link){
        // no action inside loop
    }
    return tail;
do not dereference the null pointer - only use whatever.link if node variable is not null