Thursday, May 3

No Warmup
   Remember that the questions and most answers are online at
   http://cs.millersville.edu/~ekatz/cs162/warmup.html

binarytree assignment due Friday
   due Friday May 4
   as of 7am, 4 people had turned in something

final exam
   Thursday, May 10, 10:15am - 12:15pm
   in Roddy 147 (Tuesday classroom)
   there will also be a question about intellectual property

Office hours during exam week
   Tuesday 9-10
   Wednesday 9-11
   Thursday 9-10

revisit second test and its topics
   mean 74; median 73
   most common problem was not drawing picture details
      show ALL instance variables
      I did not count off all the marked points on question 4
   when writing code,
      think about what you need to do
      one of you wrote out what needed to be done before writing code
         that's good; take time to think about what to do
   you should not feel rushed
      take your time
      check your work - does it make sense?

#3 contains
   // return whether or not target is in list with head as its start
   public static<E> boolean contains( Node<E> head, E target) {
      while (head != null) {
         if (head.getData().equals(target)) {
            return true;
         }
         head = head.getLink();
      }
      return false;
   }
#5 duplicate

```java
public void duplicate() {
    if (cursor == null) { // !isCurrent()
        throw IllegalStateException("No current item");
    } else {
        precursor = cursor;
        cursor.setLink(new Node<E>(cursor.getData(), cursor.getLink()));
        if (tail == cursor) {
            tail = cursor.getLink();
        }
        cursor = cursor.getLink();
        manyNodes++;
    }
}
```

#6 queue with array does NOT move elements except on expanding array

when array needs to expand, double size and add 1

<table>
<thead>
<tr>
<th>data</th>
<th>37</th>
<th>14</th>
<th>81</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

front 0
rear 3
manyItems 4

after removing three

<table>
<thead>
<tr>
<th>data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

front 3
rear 2
manyItems 4

but now need more space,
so double capacity and add 1 (4+4+1 = 9)
and copy values into new space starting at data[0].

Keep adding items
And then remove that first item
(but that just changes index and doesn’t change the array element)

<table>
<thead>
<tr>
<th>data</th>
<th>13</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

front 1
rear 5
manyItems 5
#7b. pop returns data from the top item which is the most accessible
should not have a loop
// returns top item while removing it
public E pop() {
    if (top == null) {
        throw StackUnderflow("Empty stack; nothing to return.");
    }
    E theData = top.getData();
    top = top.getLink();
    manyItems--; // optional
    return theData;
}

final review
if you don’t understand a question, ask me about it
it won’t look exactly like the sample
draw picture details
show ALL instance variables

    // #3 print s n times
    public static void printMany(String s, int n) {
        if (n >= 1) {
            System.out.print(s);
            printMany(s, n-1);
        }
    }

    // #4a print pattern for big value
    public static void pattern(int big) {
        for (int i = big; i >= 1; i--) {
            printMany(String.valueOf(i), i);
            System.out.println( );
        }
        for (int i = 2; i <= big; ++) {
            printMany(String.valueOf(i), i);
            System.out.println( );
        }
    }

    // #4b print pattern for big value
    public static void pattern(int big) {
        if (big > 1) {
            printMany(String.valueOf(big), big);
            System.out.println( );
            pattern(big-1);
            printMany(String.valueOf(big), big);
            System.out.println( );
        } else if (big == 1) {
            System.out.println("1");
        }
    }

    // #5 iterative count nodes in linked list
public static int listLength(Node<E> head) {
    int count = 0;
    while (head != null) {
        count++;
        head = head.getLink();
    }
    return count;
}

// #5 recursive count nodes in linked list
public static int listLength(Node<E> head) {
    if (head == null) {
        return 0;
    } else {
        return 1 + listLength(head.getLink());
    }
}

#6

stack implemented with a linked list

queue implemented with an array

#7 a. No. f is in right subtree of w (it should be further to the left in tree)
B. Inorder: bhkmorwsf. (Root in the middle)
C. Postorder: bkmhfswro (root at end)
#8

Insert: fbbasewtvx

```
// #8b recursive preorder print
public static void preorder(BTnode<E> root) {
    if (root != null) {
        System.out.print(root.getData());
        preorder(root.getLeft());
        preorder(root.getRight());
    }
}
```

```
// #8d recursive pathTo
public static String pathTo(BTnode<Character> T, char target) {
    if (T == null) {
        return "";
    } else {
        if (T.getData().equals(target)) {
            return String.valueOf(target);
        } else {
            if (target < T.getData()) {
                return T.getData() + pathTo(T.getLeft(), target);
            } else {
                return T.getData() + pathTo(T.getRight(), target);
            }
        }
    }
}
```