1 Basic Operations (Spring 2015 MT2 Q1)

a. **To the right of the BST below**, draw a BST that results if we delete 20 from the BST. You should use the deletion procedure discussed in class (i.e. no more than 4 references should change).

```
      20
     / \
   15   29
  / \   / \
17  25 35  
|    |    |
16  23 28  
  |    |    |
  18  24 33
```

b. **To the right of the minHeap below**, draw the minHeap that results if we delete the smallest item from the minHeap.

```
     0
    /|
   1 4
  / \
 8  2 6  8
```

d. Draw a valid Weighted Quick Union object that results after the following calls to connect: `connect(1, 4), connect(2, 3), connect(1, 3), connect(5, 1)`. Don’t worry about the order of the arguments to each connect call, we’ll accept any reasonable convention.
a. Show the left-leaning red-black tree that corresponds to the (2,3) tree on the left. Indicate red nodes with an asterisk (as in part (b) below).

(2, 3) tree

<table>
<thead>
<tr>
<th>10 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 8</td>
</tr>
<tr>
<td>3 6</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>1*</td>
</tr>
<tr>
<td>8*</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>95</td>
</tr>
</tbody>
</table>

Left-Leaning Red-Black Tree

b. Show the (2,4) tree that corresponds to the red-black tree on the left. Red nodes are marked with an asterisk.

Red-Black Tree

<table>
<thead>
<tr>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>1*</td>
</tr>
<tr>
<td>8*</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>12*</td>
</tr>
</tbody>
</table>

(2, 4) tree
Fill in a method, `Tree::flipHorizontally`, which should flip a symmetric binary tree’s values destructively about the root in linear time. Some helper methods (`swapNumbers` and `safePush`) are given. You may not define your own helper methods. See the example:

```java
public class Tree {
    private TreeNode root;

    private static class TreeNode {
        private int num;
        private TreeNode left, right;

        private TreeNode(int num, TreeNode left, TreeNode right) {
            this.num = num;
            this.left = left;
            this.right = right;
        }
    }

    private static void swapNumbers(TreeNode t1, TreeNode t2) {
        int temp = t1.num;
        t1.num = t2.num;
        t2.num = temp;
    }

    private static void safePush(TreeNode t, Stack<TreeNode> s) {
        if (t != null) {
            s.push(t);
        }
    }

    // Continues on next page
```
public void flipHorizontally() {
    while (____________________________________________________________) {
        ______________________________________________________________
    }
}