**Objectives:** You will gain experience with AVL put implementation

**To start the lab:** Download and unzip the file:  [http://www.cs.uni.edu/~fienup/cs1520f19/labs/lab10.zip](http://www.cs.uni.edu/~fienup/cs1520f19/labs/lab10.zip)

**Part A:** Starting with an empty AVL tree, what would be the shape of the AVL tree be after put’s for keys: 90, 60, 50, 55, 40, and 53?  (Show all necessary rotation(s) needed for each put.)

**Part B:** In lecture 23 we discussed the AVL tree rotateLeft method. For Part B, you need to implement the rotateRight method. Start by copying the rotateLeft method code, and paste it as the starting point for rotateRight. Now, modify the pasted rotateRight code is two steps:

1) updating the “pointers” to the nodes to do the right-rotation
   - HINT: Since rotateRight is a mirror image of rotateLeft, change all the left’s to right’s, and all the right’s to left’s

2) updating the balanceFactors for the rotRoot and newRoot nodes. You will need to use math similar to lecture 23 where were calculated values for the rotateLeft method. Use the next two pages to calculate needed balanceFactors for the rotateRight method. Remember the follow rules of algebra:

**Algebra Review:**
- \( a - (b - c) \) when removing the paretheses you get:  \( a - (b - c) = a - b + c \)
- \( \max(x, y) + c = \max(x + c, y + c) \) should be clear from the following diagram:

<table>
<thead>
<tr>
<th>Consider max(x, y) + c:</th>
<th>\begin{align*} \max(x, y) &amp;= y \ y + c \ 0 &amp; x &amp; y &amp; \max(x, y) + c = y + c \end{align*}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider max(x+c, y+c):</td>
<td>\begin{align*} x + c &amp; \Rightarrow y + c \ 0 &amp; x &amp; y &amp; \max(x+c, y+c) = y + c \end{align*}</td>
</tr>
</tbody>
</table>

- \( \min(x, y) + c = \min(x + c, y + c) \) similarly

- \( -\max(x, y) = +\min(-x , -y) \) should be clear from the following diagram:

| Clearly, \( \min(x, y) = x \) | \begin{align*} -y & \Rightarrow -x \\ -x & \Rightarrow 0 \\ x & \Rightarrow y \end{align*} |

\( \max(-x, -y) = -x \) and negating both sides gives:  
\( -\max(-x, -y) = -(x) = x \), so \( -\max(-x, -y) = \min(x, y) \)

- \( -\min(x, y) = +\max(-x , -y) \) similarly
Calculate the needed balance factors for the `rotateRight` method below:

Before right rotation:

```
                    B
                   /  \  
                  D   rotRoot
                   /    
               newRoot

                      T_A
                     /   \  
                    h_A  T_C
                 /    /  \  
               T_A  h_A  h_C
```

After right rotation at pivot:

```
                        B
                       /  \  
                      D   rotRoot
                       /    
                   newRoot

                      T_A
                     /   \  
                    T_C  h_C
                 /    /  \  
               T_E  h_E
```

Consider the balance factor formulas for `rotateRight`. We know from the above diagram:

- `oldBal(B) = h_A - h_C`
- `newBal(B) = h_A - (1 + max(h_C, h_E))`
- `oldBal(D) = (1 + max(h_A, h_C)) - h_E`
- `newBal(D) = h_C - h_E`

To determine `newBal(D)`, consider:

```
newBal(D) - oldBal(D) =
```

(See back for `newBal(B)` calculation)
Consider the balance factor formulas for rotateRight. We know from the above diagram:

\[
\begin{align*}
\text{oldBal}(B) &= h_A - h_C \\
\text{oldBal}(D) &= (1 + \max(h_A, h_C)) - h_E
\end{align*}
\]

To determine \( \text{newBal}(B) \), consider:

\[
\text{newBal}(B) - \text{oldBal}(B) = \]

After completing your implementation of rotateRight, test your code by running the avl_tree.py program. Once you think it is working, run the timeAVLTree.py program. The height of AVL tree after adding in sorted order should be 13, and the height of AVL tree after adding in shuffled order should be about 15.

When it is working, raise your hand and we will check you off.

(If you have extra time, consider working on previous labs or homeworks.)