Data Structures (CS 1520) Lab 10 Name:________________________

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

To start the lab: Download and unzip the lab10.zip file eLearning.

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:

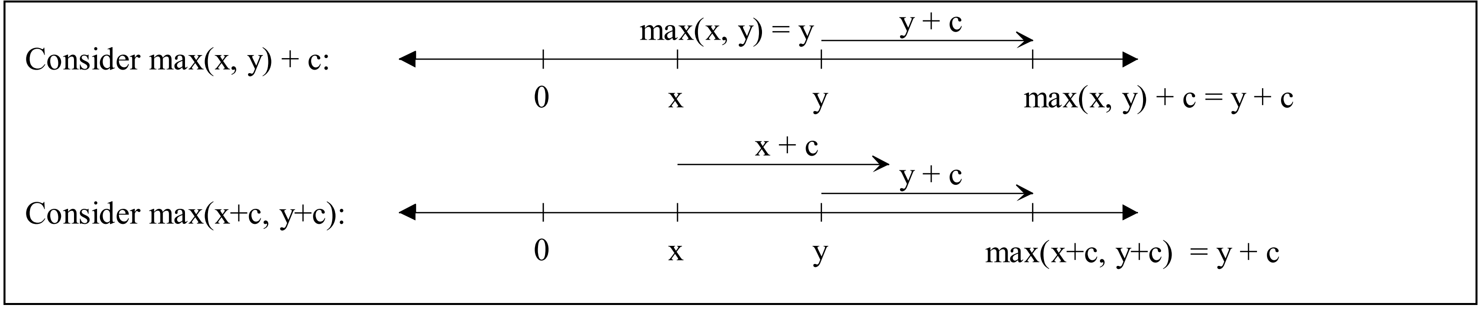
1) updating the "pointers" to the nodes to do the right-rotation

- 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

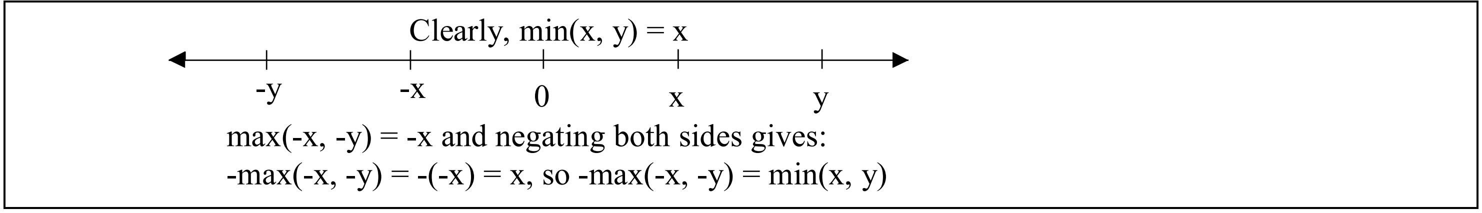
2) updating the balanceFactors for the rotRoot and newRoot nodes. You will need to use math similar to lecture 23 where were calculated values for the `rotateLeft` method. Use the next two pages to calculate needed balanceFactors for the `rotateRight` method. Remember the follow rules of algebra:

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:



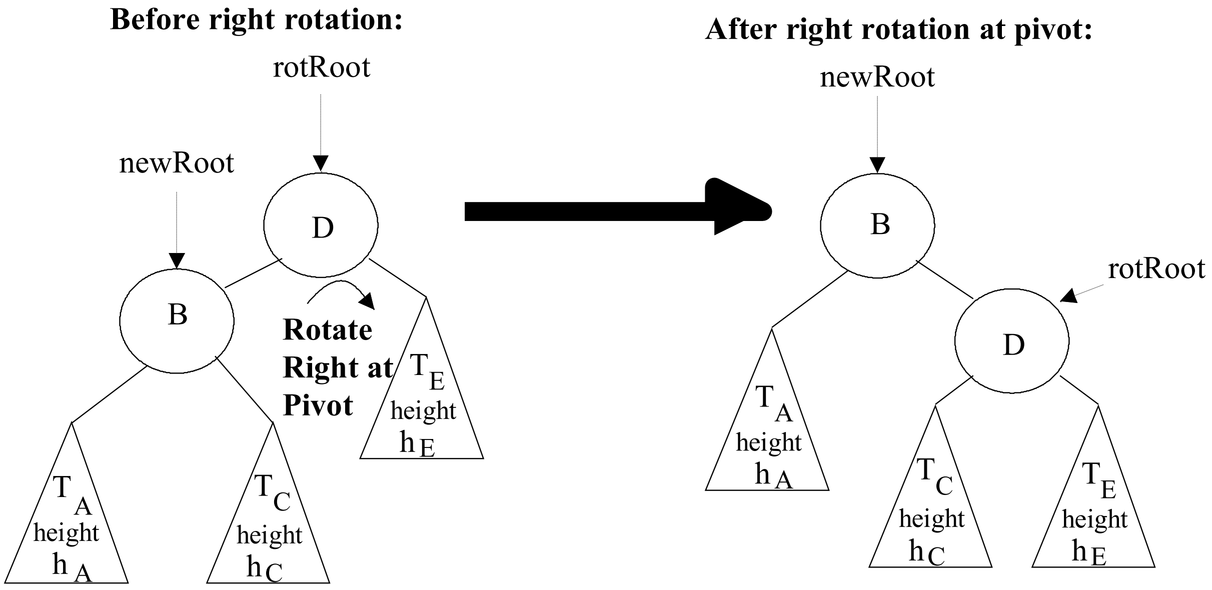
- **min(x, y) + c = min(x + c, y + c)** similarly
- **-max(x, y) = +min(-x , -y)** should be clear from the following diagram:



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 balanceFactors for the `rotateRight` method below:



Before right rotation:

After right rotation at pivot:

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

oldBal(B) = $h_A - h_C$ and

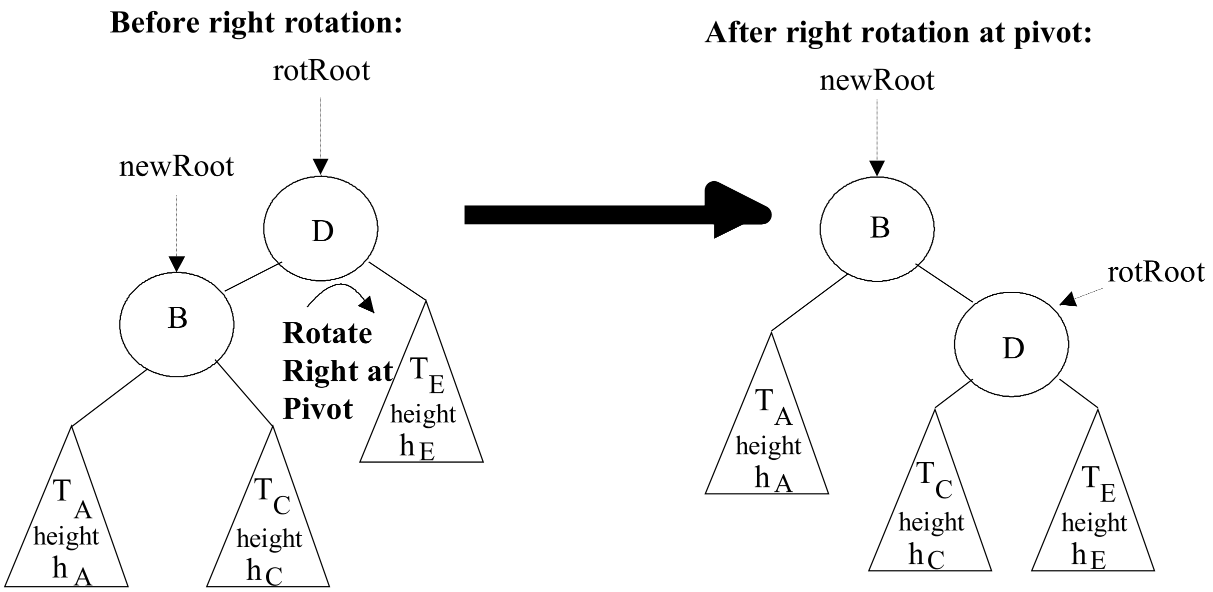
oldBal(D) = $(1+max(h_A, h_C)) - h_E$

newBal(B) = $h_A - (1 + max(h_C, h_E))$ and

newBal(D) = $h_C - h_E$

To determine *newBal(D)*, consider:

newBal(D) - oldBal(D) =



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

oldBal(B) = $h_A - h_C$ and *newBal(B)* = $h_A - (1 + max(h_C, h_E))$ and

oldBal(D) = $(1+max(h_A, h_C)) - h_E$ *newBal(D)* = $h_C - h_E$

To determine *newBal(B)*, consider:

newBal(B) - 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.**

After you have answers and correct code for all parts of the lab, submit a lab10.zip containing your code on eLearning. If you do not get done today, then submit it by next week's lab period.

Remember to save your lab10 files for later usage on homework assignments!