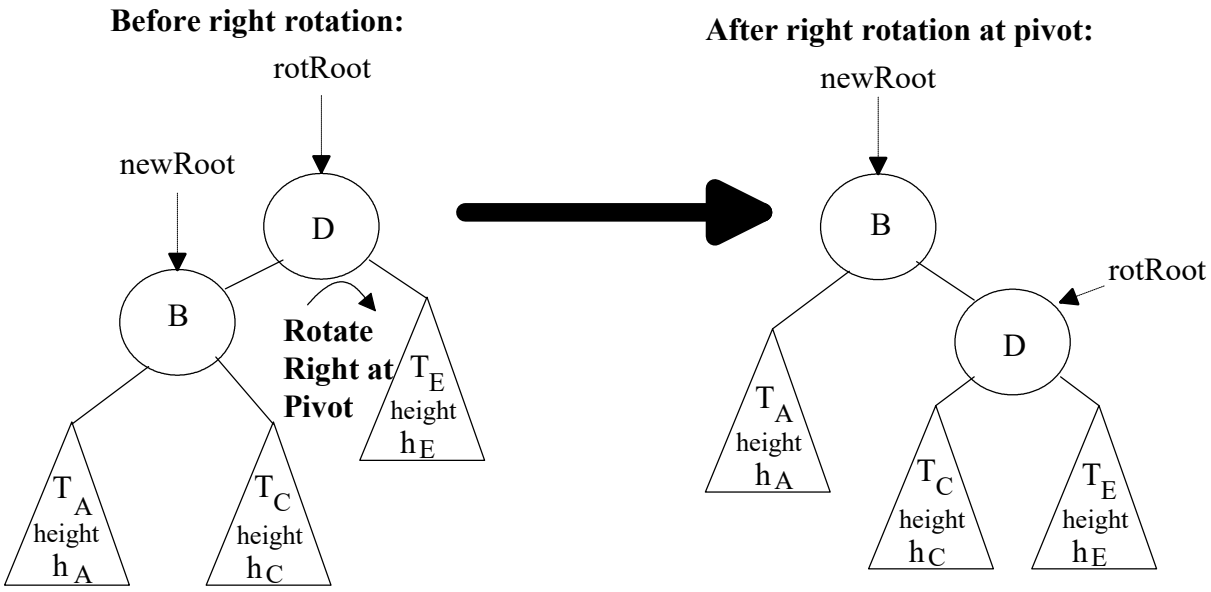
Data Structures (CS 1520) Lab 10 Name:_____

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

To start the lab: Download and unzip the file: http://www.cs.uni.edu/~fienup/cs1520s19/labs/lab10.zip

Part A: In lecture 23 we discussed the AVL tree `rotateLeft` method. For this lab you need to implement the `rotateRight` method. The `rotateRight` method has two parts:

- updating the "pointers" to the nodes to do the rotation (look at the `rotateLeft` method code)
- updating the balanceFactors for the rotRoot and newRoot nodes (you need to use math similar to lecture 23) (See below too)



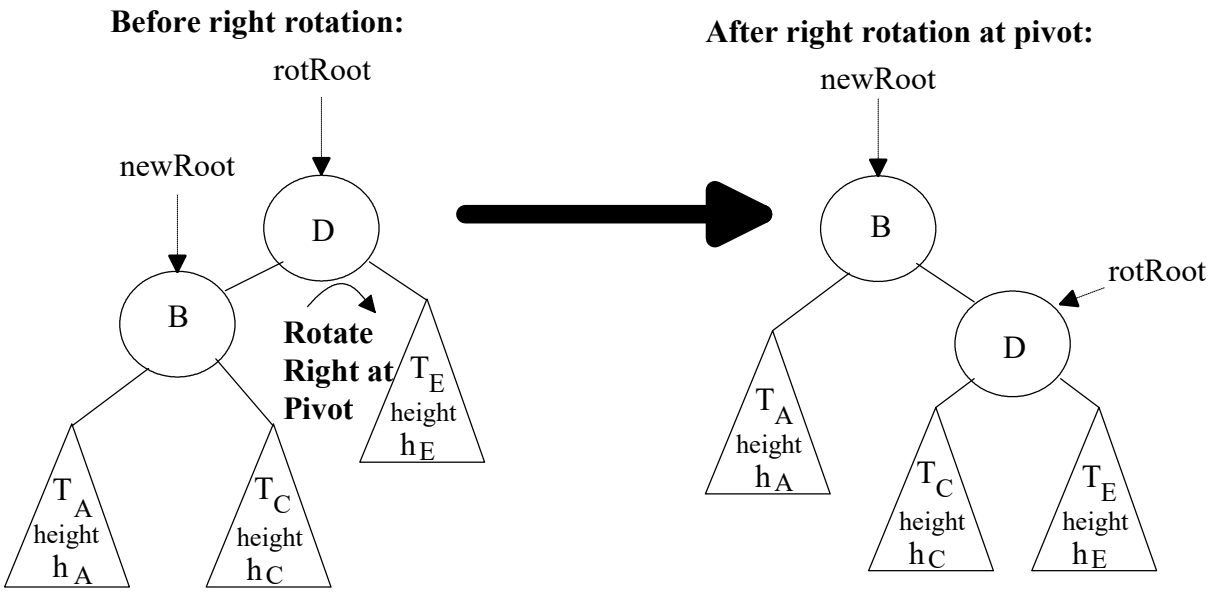
Consider the balance factor formulas for `rotateRight`.
oldBal(B) = h_A - h_C and
oldBal(D) = (1+*max*(h_A , h_C)) - h_E

We know from the above diagram:
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)* =

(See back for *newBal(B)* calculation)



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(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.

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.)