CSCI 2002 Module 4 and 5 Assignment

This assignment will be submitted via D2L. There will be a document on D2L that gives details on submitting written assignments via D2L.

Part 1 (10 points) – unbalanced binary search trees

Add the following numbers, in the order given, to a binary search tree, and draw a diagram of the resulting tree. For this part of the assignment, you need only show the final tree.

36	21	70	20	14	88	96	74
81	19	83	68	93	16	64	99

Label the height of each node in the final tree.

Part 2 (60 points) – balanced binary search trees

In this part of the assignment, you will draw four AVL trees, balancing them as you add items.

Every time you need rebalance the tree, you must label it to show

- which of the 4 cases it is (right/right, right/left, left/right, or left/left),
- which rotations (left or right) are performed.

If a single rebalancing operation requires two rotations, you may either show the end result after both rotations, or show both rotations separately.

Every time a tree is rebalanced, you should draw the resulting tree in black, and draw the newly-added nodes (up to the next rebalancing) in a different color. The last section of this assignment shows an example of what I'm looking for.

If you want to check your work for correctness, you may use this AVL Tree simulator:

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Note, however, that this does not tell you which case each rebalance operation was, nor which rotations were performed. It just shows the end product.

Exercise 1 – ascending order

Add the numbers 1 to 12 to a balanced BST in ascending order.

Exercise 2 – descending order

Add the numbers 1 to 12 to a balanced BST in descending order.

Exercise 3 – random order

Add the numbers from Part 1 of this assignment to a balanced BST in the order given.

Exercise 4 – random order

Add the following numbers, in the specified order, to a balanced BST.

12	95	63	16	27	21	69	51
85	45	83	73	24	33	28	52

Example:

The following pages show the result of adding these numbers to an unbalanced BST and a balanced AVL tree, in the format required for this assignment:

27	84	69	70	67	93	54	36
87	53	45	58	57	35	13	39

I color coded as follows:

- **Black – Nodes from the previous rebalancing operation**
- **Green – newly added node since the last rebalance**
- Purple – the unbalanced node forcing the rebalancing, along with the heights of the left and right subtree.
- Blue – the L and R designators showing the heavier child and grandchild nodes of the node requiring rebalancing.

Note that I didn't show the heights for all nodes; I only labelled the heights of the left and right subtrees of the node needing to be rebalanced.

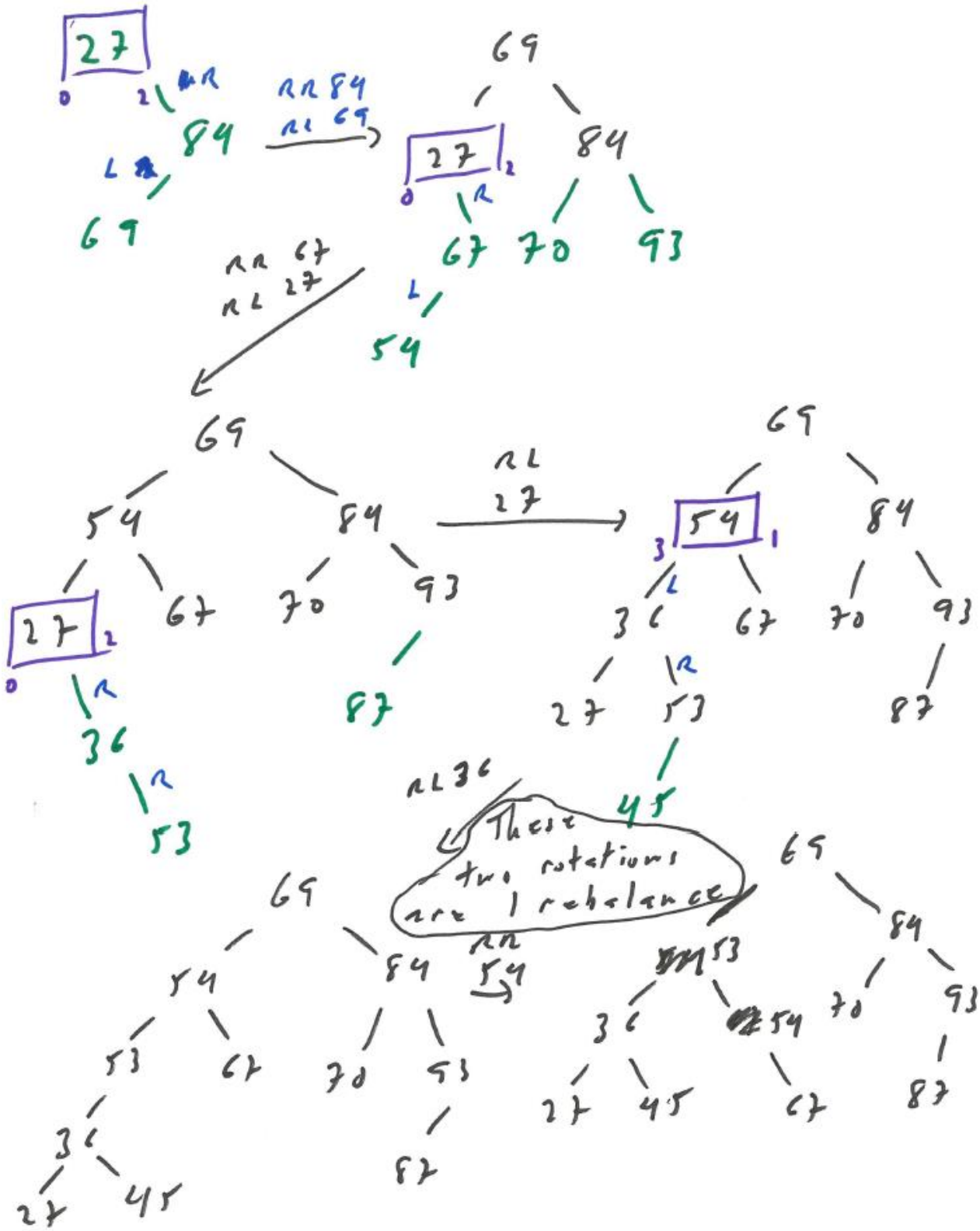
The arrows from one tree to the other show the rotations performed, using RL(N) for a left rotation around node N, and RR(N) for a right rotation around node N.

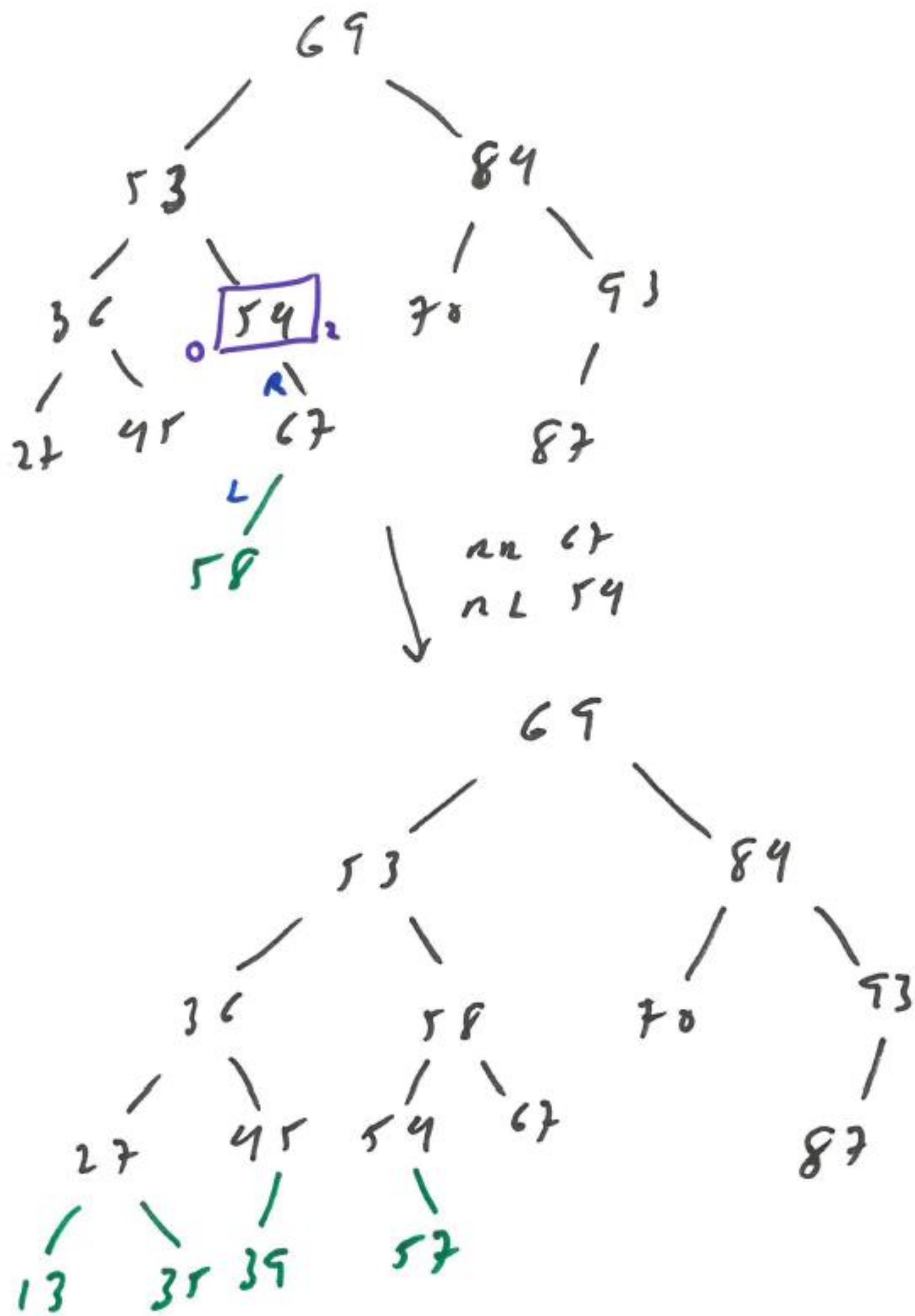
For most rebalancing operations that required two rotations, I did both rotations at once, and labelled the arrow accordingly. This is fine for rotations at the bottom of the tree. However, the rebalancing performed after adding 45 was complex enough to warrant doing it in two steps.

Everything listed **in bold** above describes a required part of your diagrams.

Note that the last 4 keys required no rebalancing. That is very normal for AVL trees; as the tree gets larger, more nodes can generally be added before a rebalancing is needed.

Also note that the final tree looks unbalanced, since there are 11 nodes to the left of the root, and 4 nodes to the right. For an AVL tree, that can happen. What counts is the height of each subtree, not the number of nodes in it.


27
84
69
RR 84
RL 69
27
84
69
67
70
93
RR 67
RL 27
54
69
54
84
67
70
93
27
36
53
87
RL
27
54
84
3
1
2
36
67
70
93
27
53
87
RL 36
45
These two rotations are 1 rebalance
69
54
84
RR 54
54
53
67
70
53
87
36
27
45
69
84
53
36
54
70
93
27
45
67
87


rR 67
rL 54