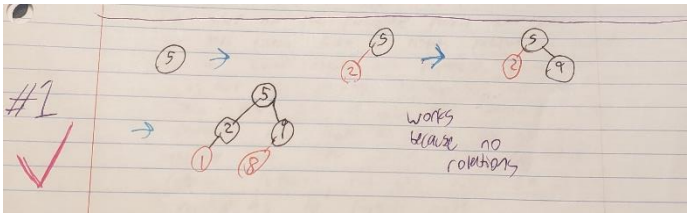
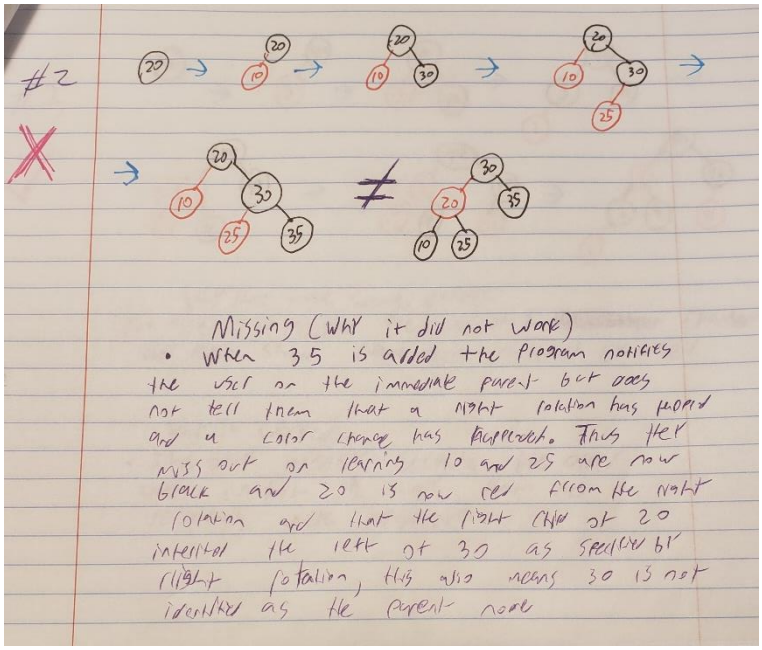
Method one works great if one understands that left and right rotations occur and are willing to account for these rotations. Example cases are tested below….

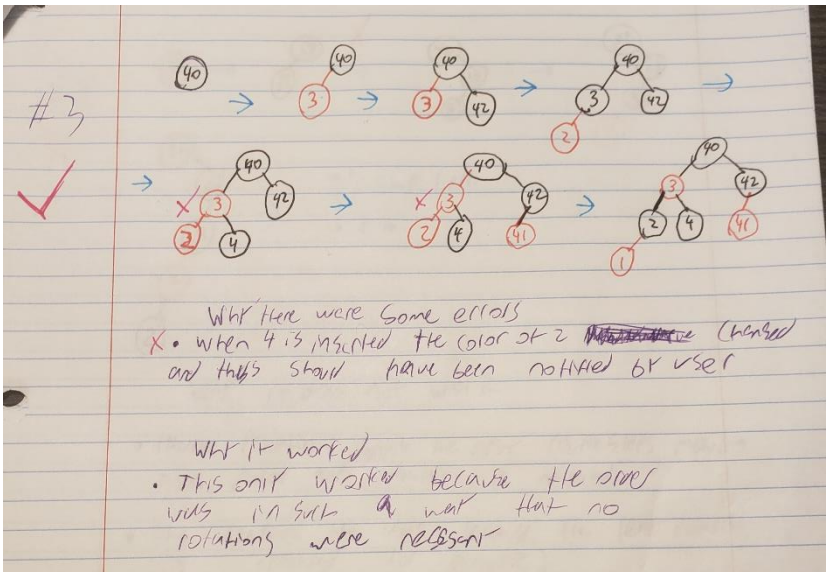
For Test Case #1 Method One Works Perfect Because there are no left or right rotations:



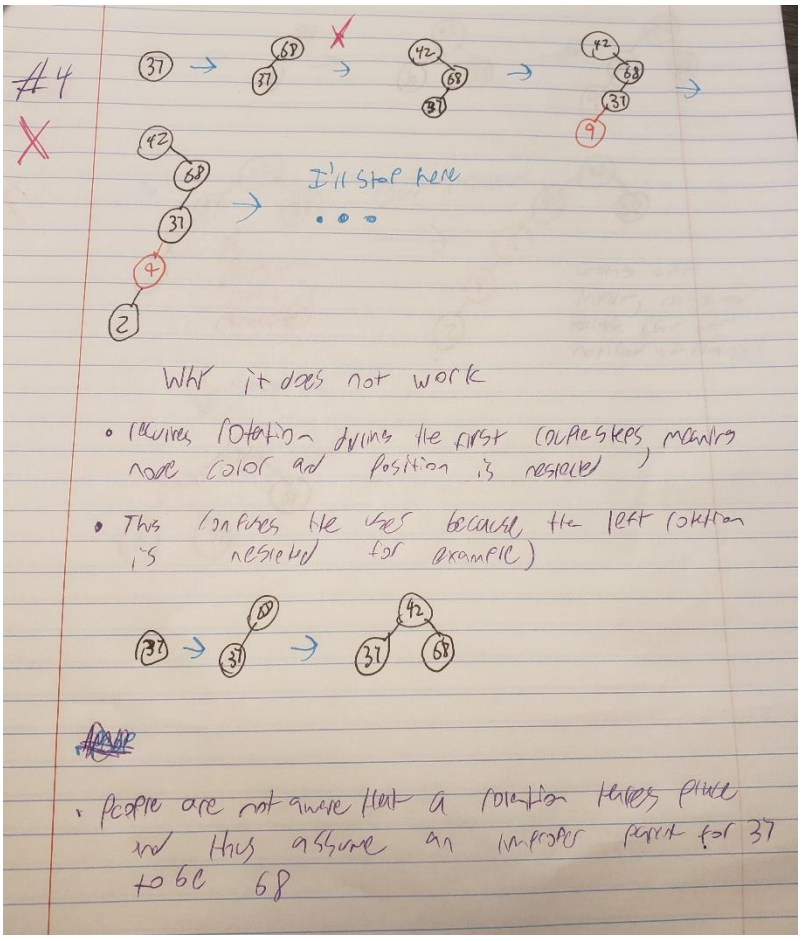
For Test Case #2 however a rotation occurs and thus the user is inaware that 30 becomes the parent and 20 inheites the left child of 30 as described by method two and as described by basic left rotation definition. This is why the code includes a show left and right rotation option so the user can understand that other nodes are shifting in the process. Please look at case 5 to see how when left and right rotations are regarded along with the desribed output the proper tree can be formed.

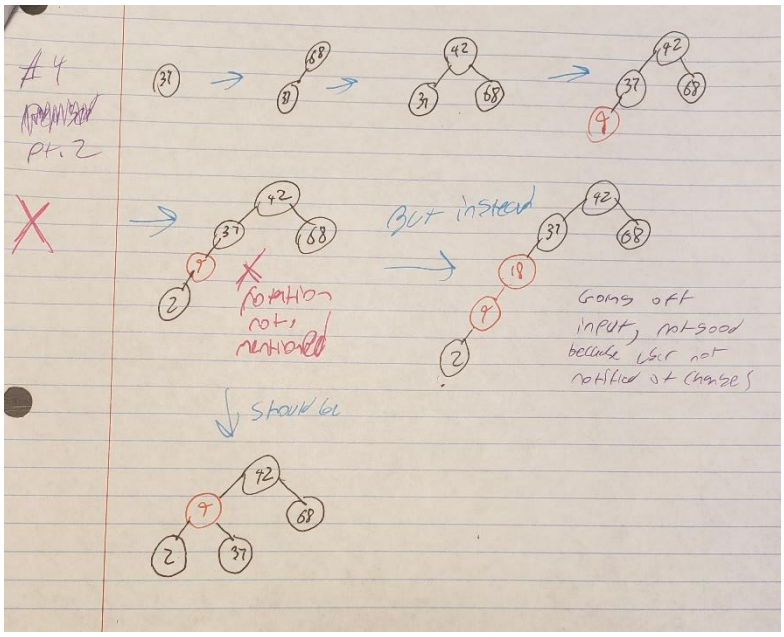


Test case #3 is a half in half, senario in that it works good in the end but because some colors are not being acknowlaged during rotations the tree is temporaraly improper with two red connecting childs. Luckly the final insertion fixes this. But keep in mind it if was just this without the final insertion than it is up to the user to interpret that a rotation occurred and thus colors were altared.

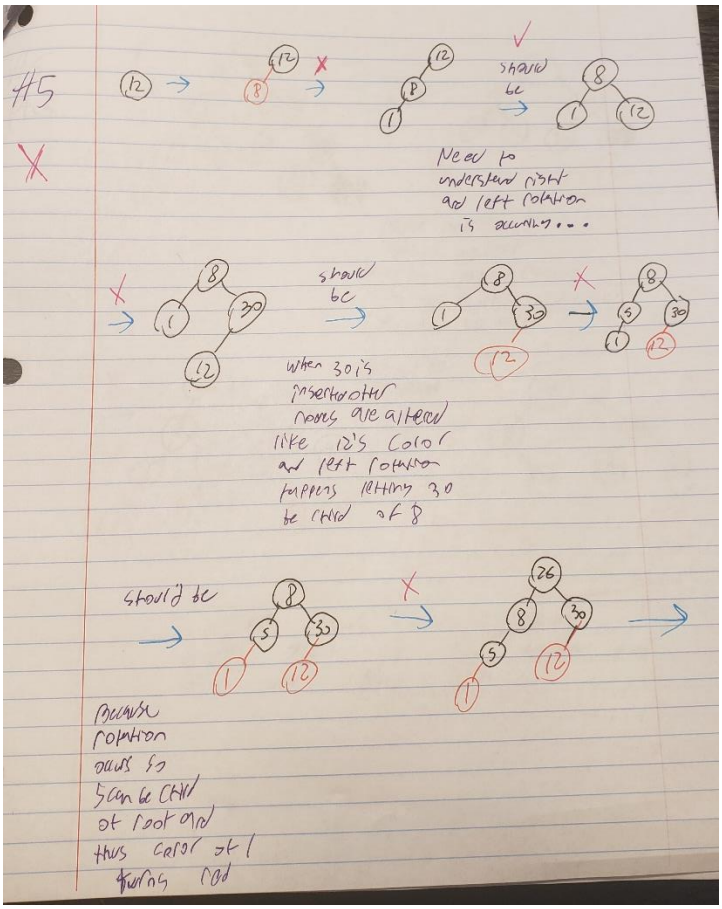


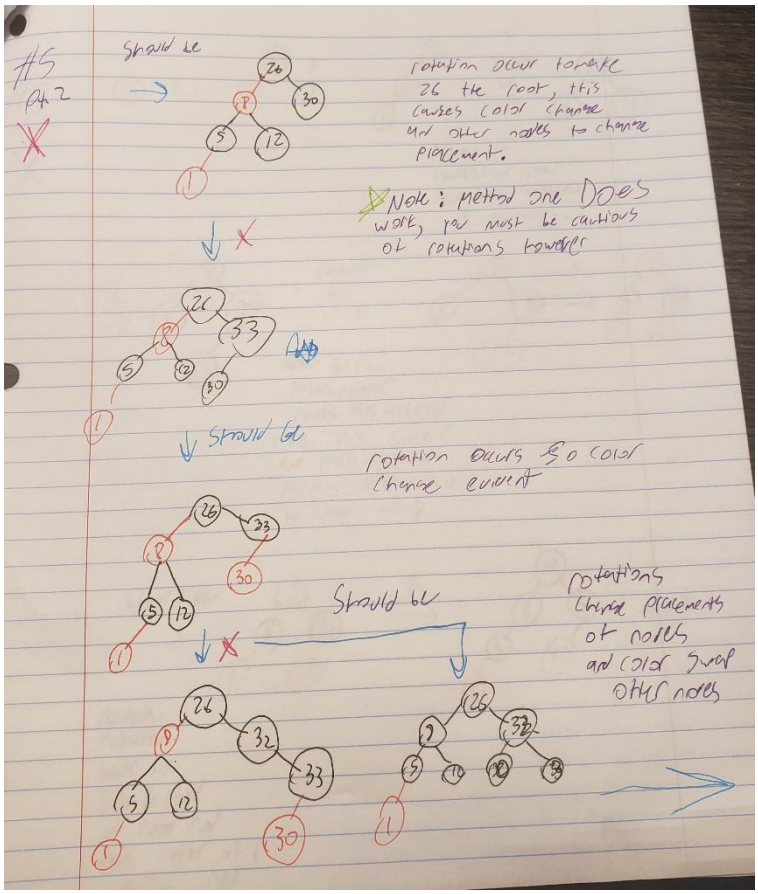
Test case #4 does not go through the whole tree but just showcases how individuals could be confused with the output if they don't understand when rotations are occuring and which rotations are occuring.

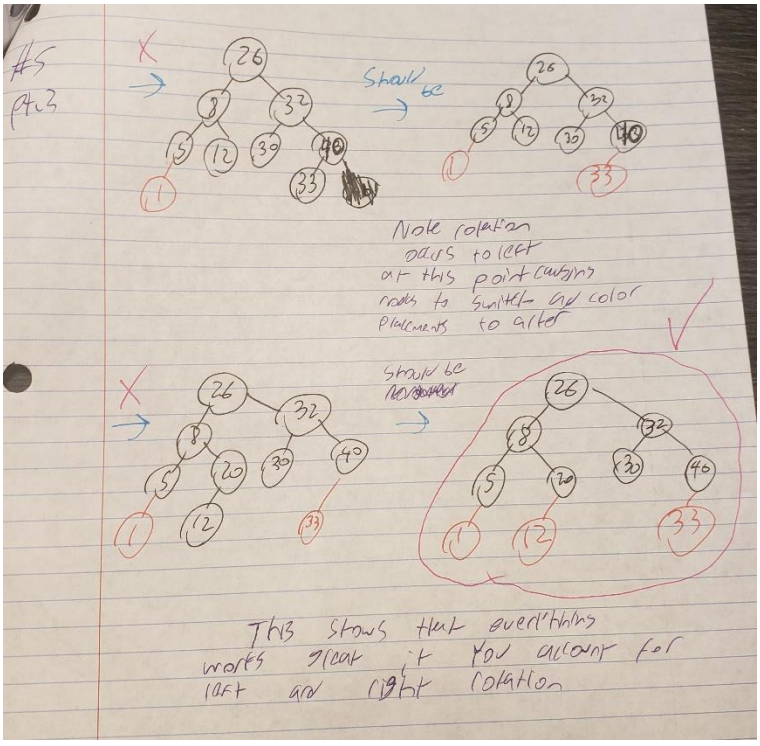




Test case #5 goes through the whole tree while showing what the user could misinterpret and also show how by getting the information of what left or right rotation is happening the proper tree can be formed. This shows that method one is appropriate for this project because it shows the process piece by piece for a LLRBT insertion, however by adding the extra information of left and right rotation it becomes evident to the user that other nodes are shifting place and changing color thus giving off the proper and correct tree. This is true for all trees meaning method one is a good indication of what is going on with the system. But if you are looking for youself I highly recommend looking at the output with rotation option set to 1 so it is easier to understand when rotations are occuring and thus indirectly understand how other nodes are changing color and placement. I could have re displayed every node after the right and left rotation however that is NOT the specifiaction of the project. However, personally if I were to do this again without project specification limitations I would output after each rotation occurs that way the user is informed on the exact updated information of each node instead of trying to understand what exactly the rotations do. However, it is super good practice to try the left and right rotations on your own without input. I feel that by not outputing data after rotations I enhanced my understanding of rotations… 😊


#5
should be
Need to understand right and left rotation is actually...
should be
When 30 is inserted other nodes are altered like it's color and left rotation happens letting 30 be child of 8
should be
Because rotation occurs so 5 can be child of root and thus color of 1 turning red





Pink circle is proper output of tree. This is checked and proved by method 2.