**Programming and Problem Solving**

**Assignment # 3**

**PART 1**

**SUBMITTED BY**

**Name:** Manan Dineshbhai Paruthi

**Student ID:** 40192620

**Question 1**

1. Mark the removed entry location as the negative key of the removed value instead of marking it as Available

* No advantages of this proposal
* There will be no impact / difference in time and space complexity

1. Instead of using AVAILABLE, find a key in the table that should have been placed in the location of the removed entry, then place that key (the entire entry of course) in that location (instead of setting the location as AVAILABLE)

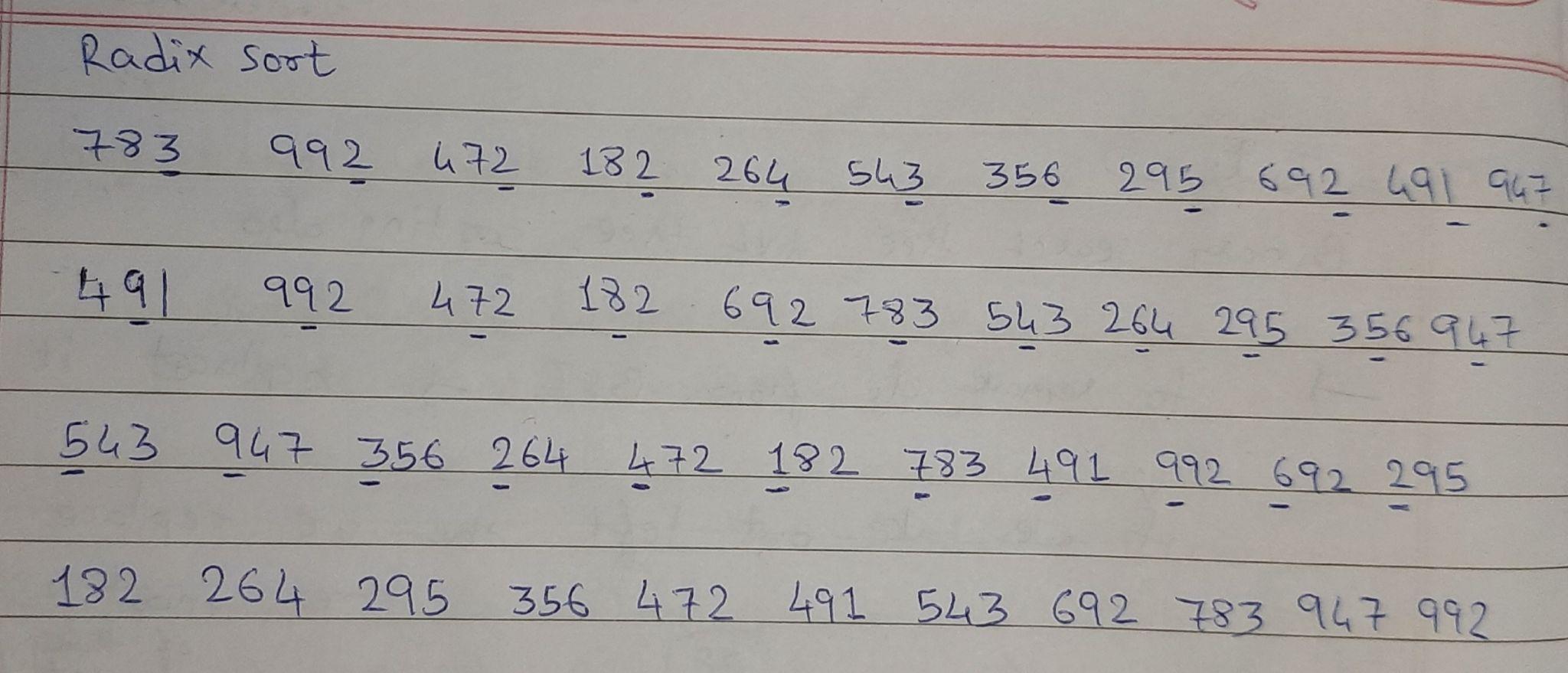
* Advantage of this proposal is searching / retrieval of key will be faster - strict O(1)
* But the time complexity of removal will degrade from O(1) to O(n) as we’ll need to generate a key again from hash function O(1) & find that key in HashTable O(n) to replace it in the correct hashed index position
* Example :

Let’s assume the hashtable is of length 9

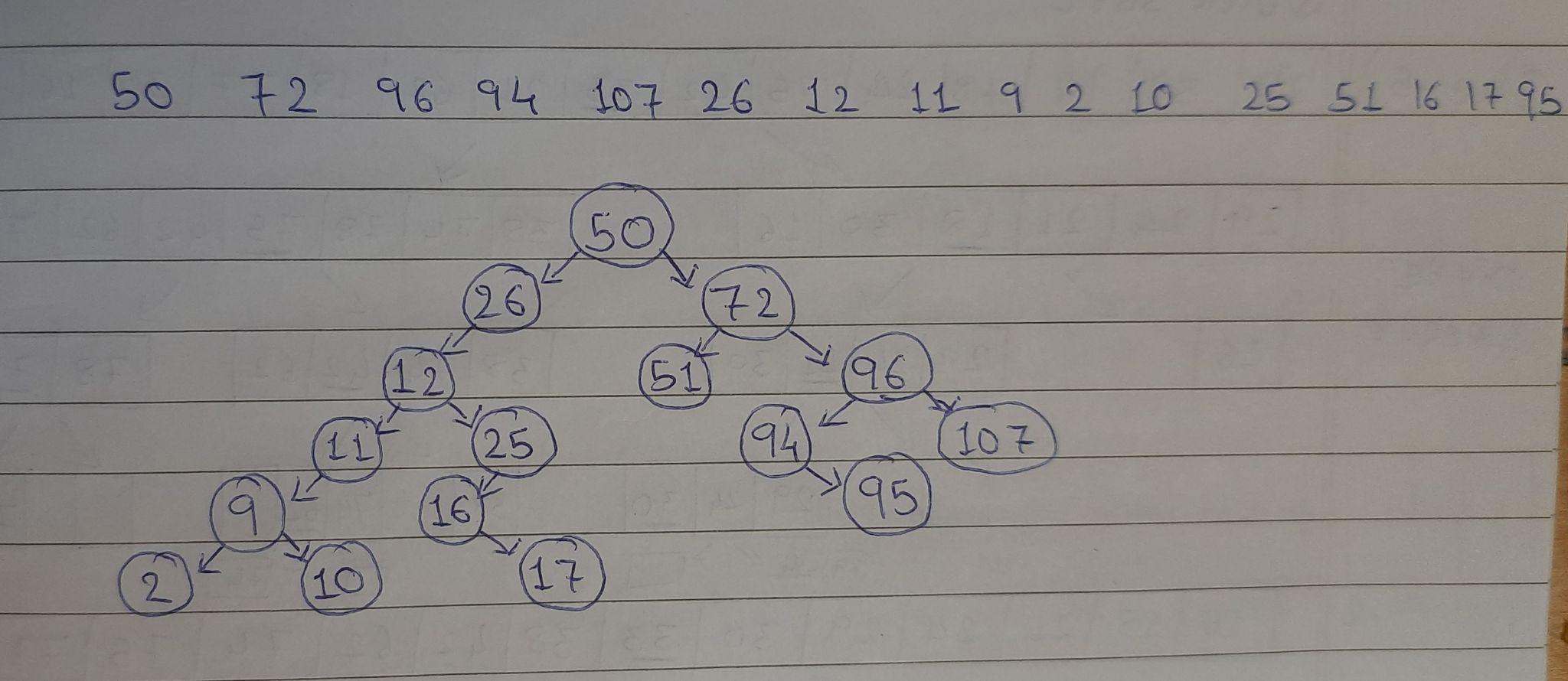
In Linear Probing, the actual hashIndex of key X was 2 but due to collision it was inserted at 8th index as other indexes from 3rd to 7th were full

Then when the value at index 2 was removed and we are trying to find the key which was supposed to be at that index it will take O(n) time complexity as we’ll need to traverse from 3rd to 8th index and then replace it at its correct position

**Question 2**

****

**Question 3**

****

**Pseudo Code - Question 4**

**findHeight(node T)**

**IF** T == null **THEN**

**RETURN** 0

**END-IF**

**IF** T.balance >= 0 **THEN**

**RETURN** 1+ findheight(T.left)

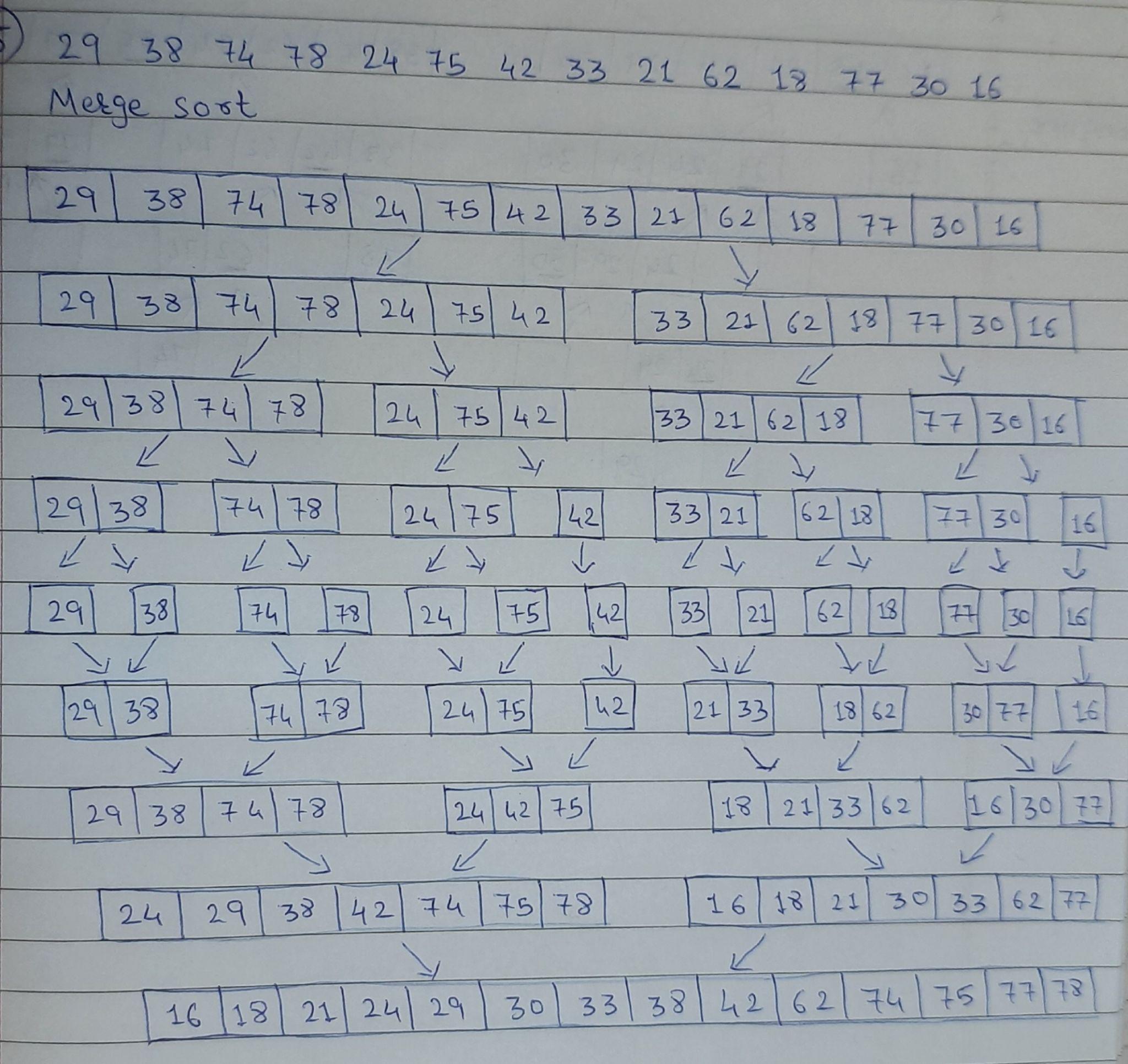
**ELSE**

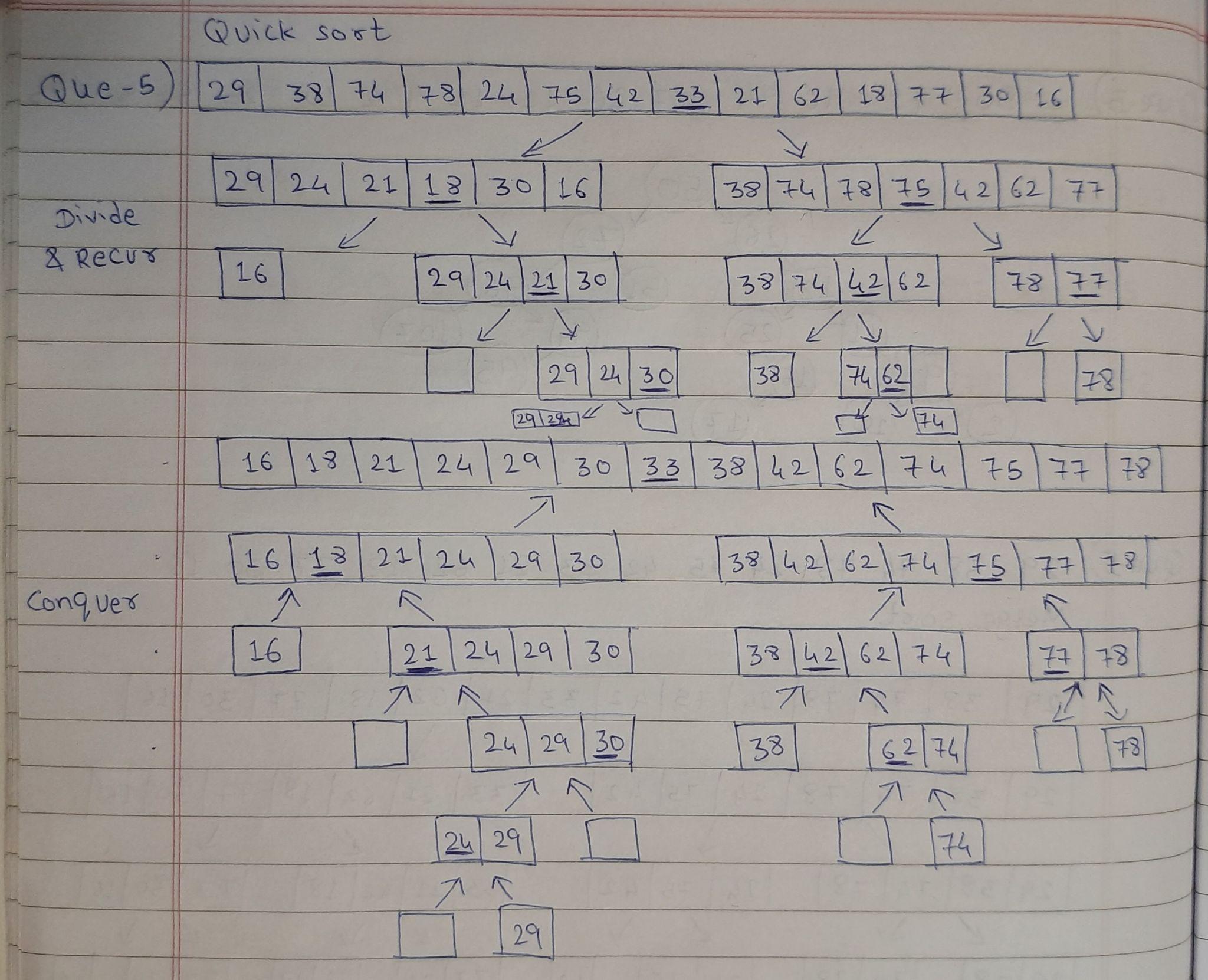
**RETURN** 1 + findHeight(T.right)

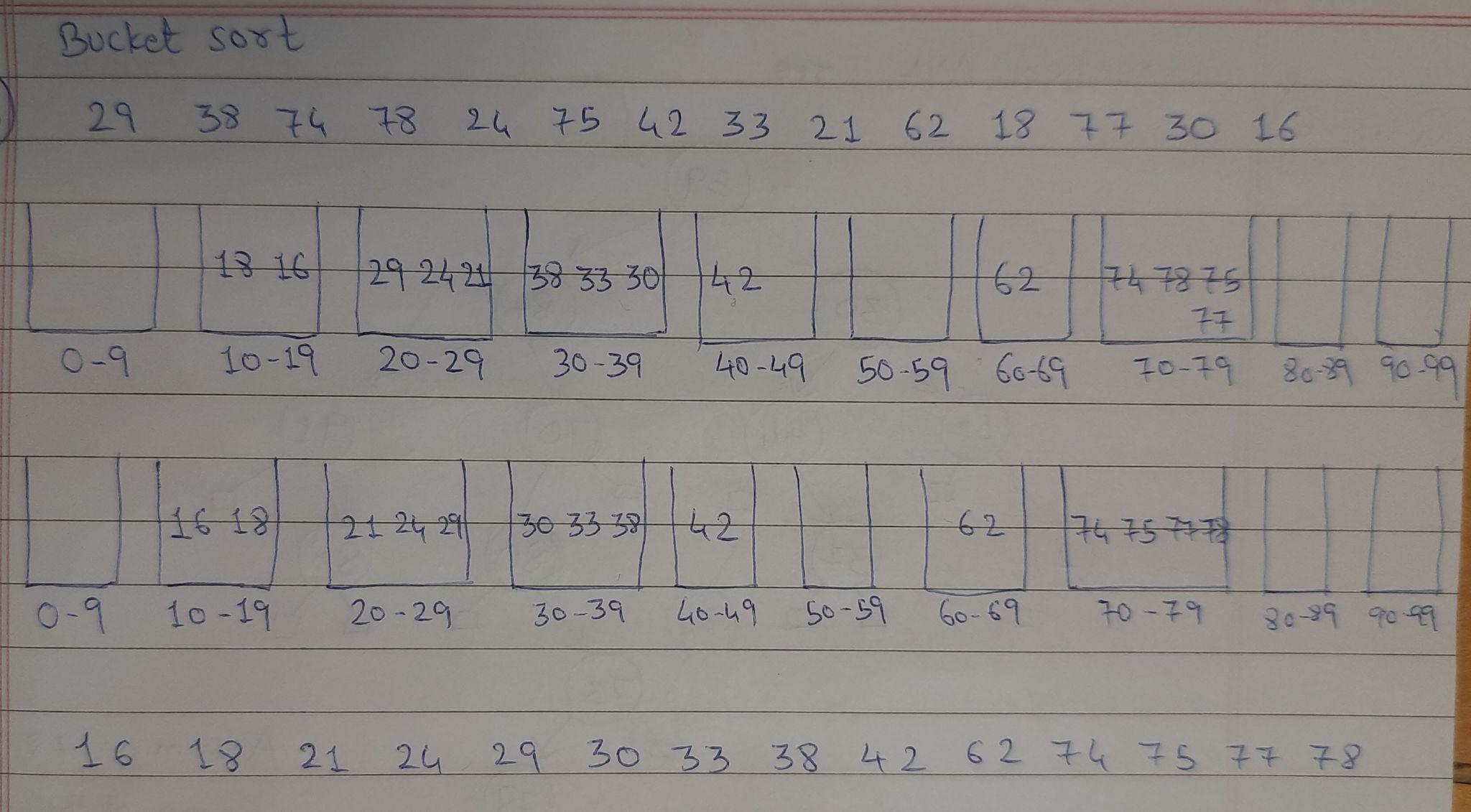
**END-IF**

Algorithm - balance is height(left) - height(right) so if T.balance is greater than 0 then left tree is largest and if T.balance is less than 0 then right tree is largest. So based on balance factor we know which side of tree is largest so we will be traversing only to that place and it happens in recursion so time and space complexity both are O(log n)

**Question 5**

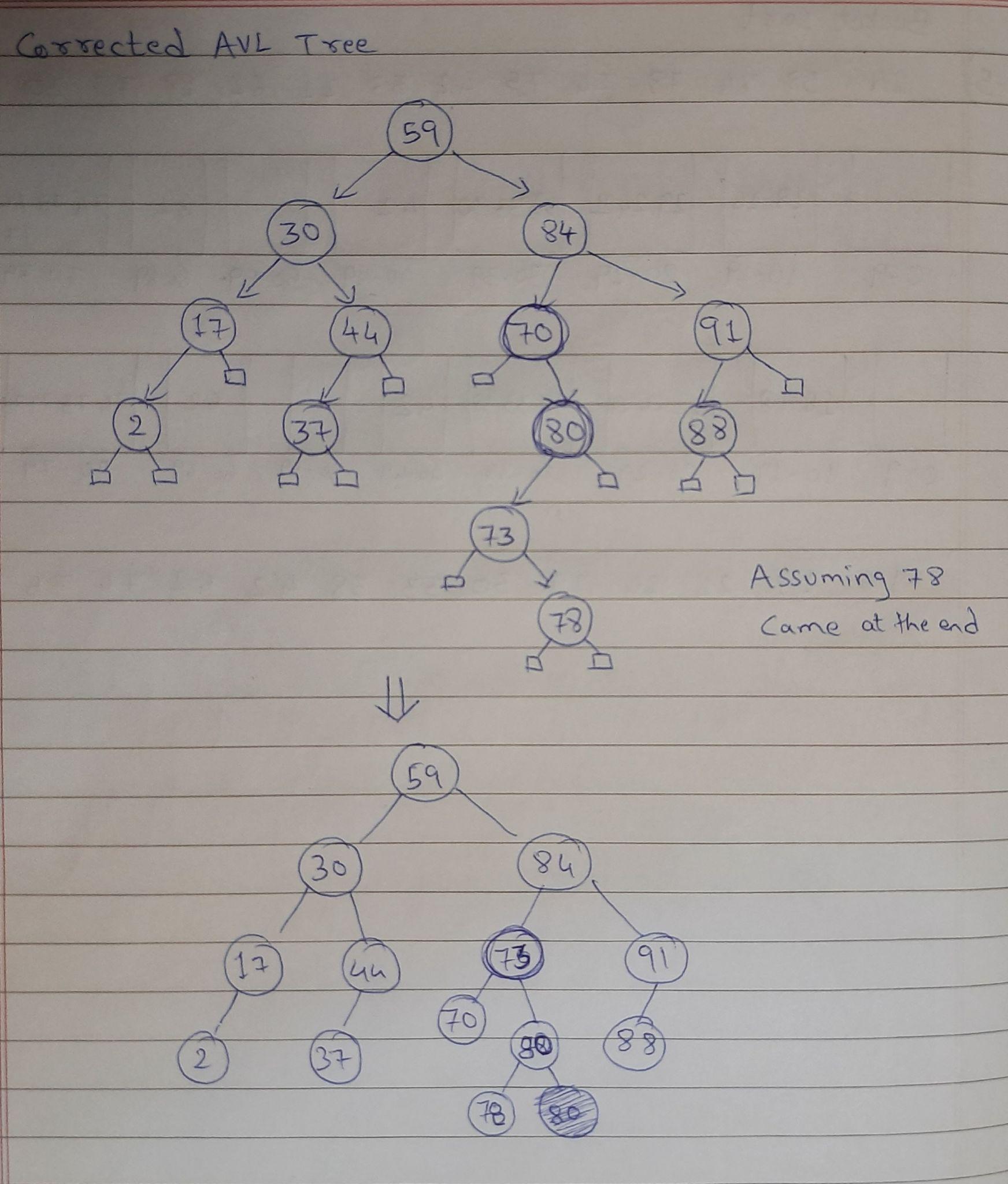




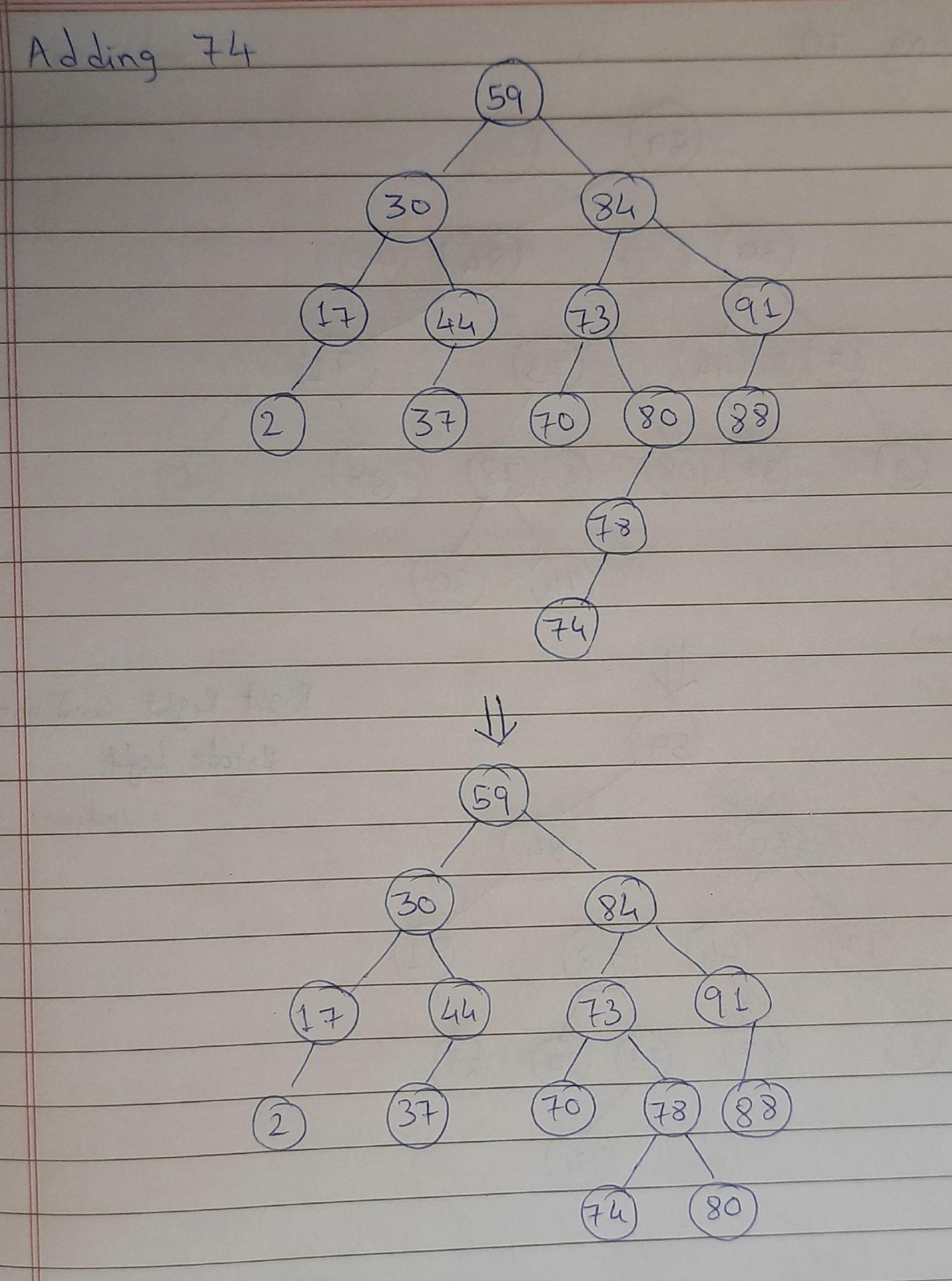


**Question 6**

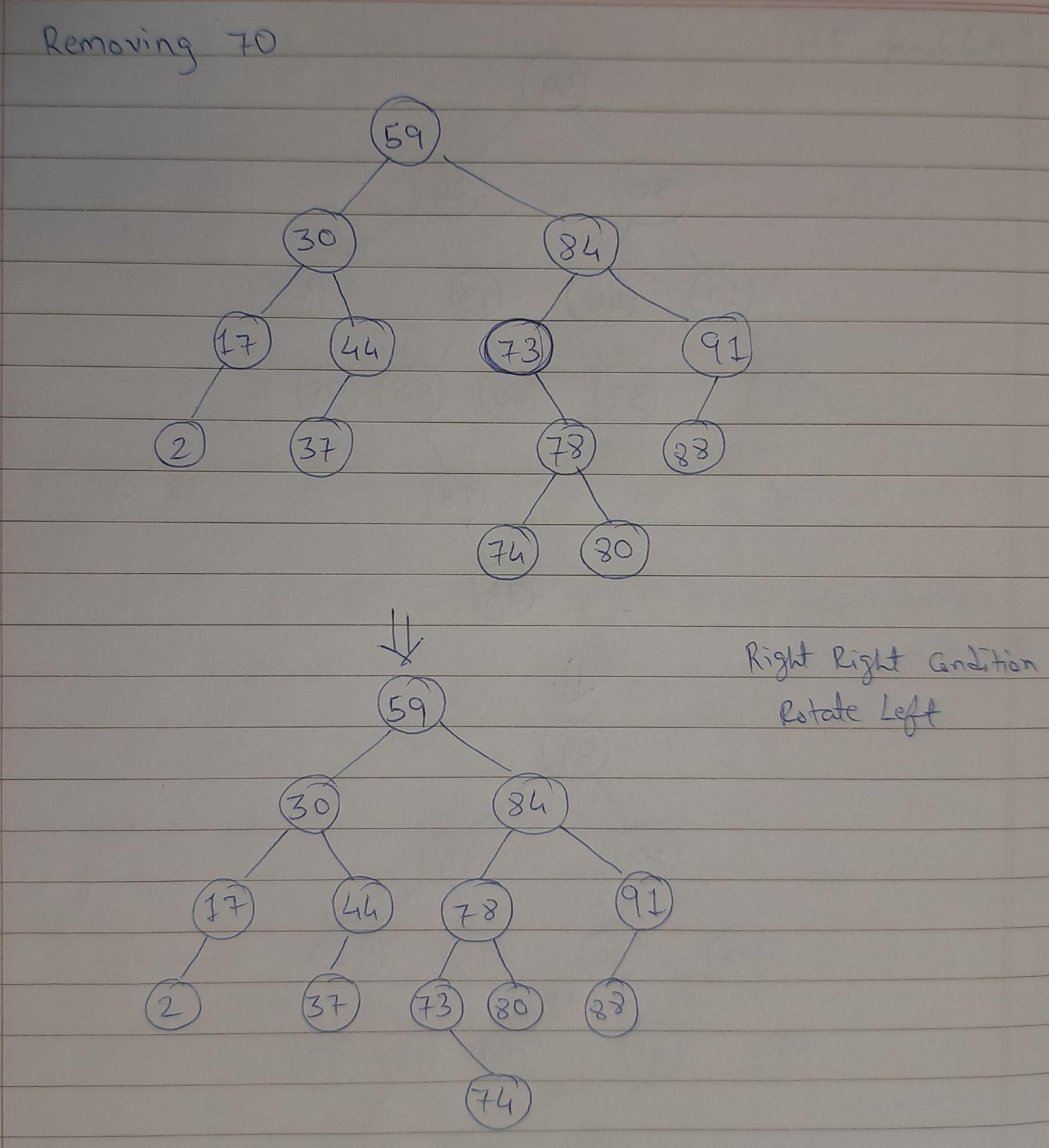
1. Errors in the given AVL tree - 2 is in incorrect position & nodes with value 70 & 80 are imbalanced

****

1. Time Complexity of adding 74 is O(log n)



1. Time Complexity of removing 70 is O(log n)



1. Time Complexity of removing 91 is O(log n)

