# LAB 09

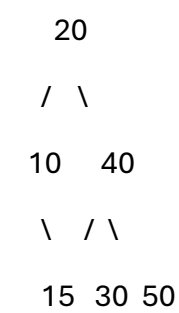
## TASK 0

AVL trees automatically rebalance themselves to maintain minimum height. This ensures faster search, insertion, and deletion with guaranteed **O(log n)** time.

To convert the given BSTs into AVL trees, the code first stores all nodes using inorder traversal, which produces a sorted list of values. Then the AVL is built by recursively choosing the middle element of the array as the root, linking the left half as the left subtree and the right half as the right subtree, ensuring perfect balance. In cases where a rotation is needed, temporary pointers are used: **x** is the unbalanced node, **y** is its child causing the imbalance, and **t2** is the subtree that gets repositioned during rotation; in BST A this corresponds to a left-left imbalance requiring a **right rotation**, and in BST B a right-right imbalance requiring a **left rotation**. The function finally returns the node created from the middle element because it becomes the new balanced root of the AVL tree, guaranteeing minimal height and equal distribution of nodes on both sides, which is why this node must be returned as the final AVL root.

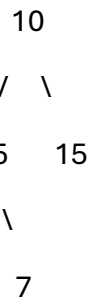
## TASK 1

To insert the new student with roll number **15**, we first compare it with the AVL tree values. Since 15 is greater than 10, it moves to the right, but less than 20, it becomes the left child of 20. The insertion causes an imbalance at node 10 because the right subtree becomes heavier, producing a **Right-Left imbalance (RL case)**. To fix this, a **right rotation** is first performed on node 20, followed by a **left rotation** on node 10. After balancing, the height of the AVL tree becomes **3**, as the root now has two balanced subtrees of height 2 and 1. The final AVL:



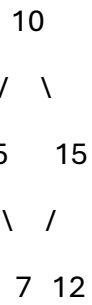
## TASK 3

**1. Initial AVL Tree (Before inserting 12)**




Height = 3
Tree is balanced.

**2. Insert 12**




**3. Balance Factor of Each Node**

| Node | Left Height | Right Height | BF |
|---|---|---|---|
| 3 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 |
| 5 | 1 | 1 | 0 |
| 15 | 1 | 0 | +1 |
| 10 | 2 | 2 | 0 |

**4. Check if Tree is Unbalanced**

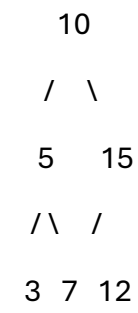
All balance factors are between **-1 and +1**
**Tree remains balanced**

**No rotation is required.**

**5. Final Height of the Tree**

Height = **3**

**6. Final Balanced AVL Tree**




Everything stays balanced.