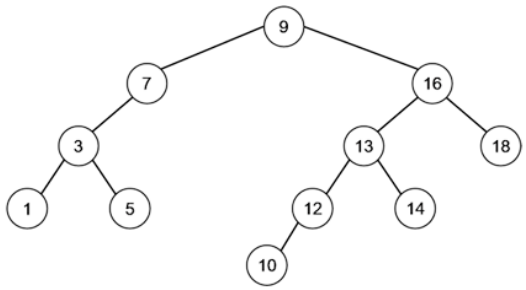
University of Colombo School of Computing
SCS 1308 - Foundations of Algorithms

Tutorial -13

01. In the following tree, which nodes violate the height invariant



---

02. Draw all the rotations that you must perform and the final AVL tree after the following elements are inserted in the given order starting from an empty tree

1, 10, 5, 7, 3, 13, 6, 4, 8, 9

---

03. Define an AVLTreeNode structure with:

- ➔ Integer key
- ➔ Integer height
- ➔ Pointers to left and right child nodes.

04. Implement a function to create a new AVL tree node.

A. Implement the function getHeight(AVLTreeNode* node) to return the height of a node.

B. Implement the function getBalanceFactor(AVLTreeNode* node) to return the balance factor.

C. Insert nodes {10, 20, 30} into the AVL tree.

D. What will be the balance factor of the root node after inserting 30?

   Does the AVL condition hold? Why or why not?

05. Implement the function ***rightRotate(AVLTreeNode* y)*** (LL Rotation).

06. Implement the function ***leftRotate(AVLTreeNode* x)*** (RR Rotation)

07. Implement ***leftRightRotate()*** (LR Rotation).

08. Implement ***rightLeftRotate()*** (RL Rotation).

09. Insert nodes `{10, 20, 30}` and observe that a **Right Rotation (LL Rotation)** is needed.

10. Draw the tree structure before and after the rotation. Explain how the rotation restores balance.

11. Modify the insertion function to include rotation operations when an imbalance occurs.

    A. Insert the following nodes step by step: `{10, 20, 30, 25, 27, 7, 4, 23, 26, 21}`.

    B. Draw the AVL tree after each insertion.

    C. Identify the first node where an imbalance occurs.

    D. Determine the required rotations to restore balance at each step.