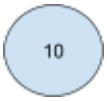
H01

Set 1: Left-Right and Right-Left Rotations Insert the following values in order: 10, 20, 30, 15, 25, 5, 35, 27

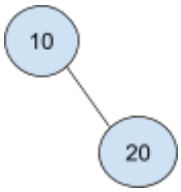
1. Insert 10:

- The tree is empty initially, so 10 becomes the root.



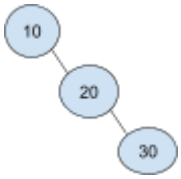
2. Insert 20:

- 20 is greater than 10, so 20 is inserted as the right child of 10.



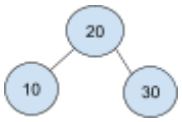
3. Insert 30:

- 30 is greater than 10 and 20, so 30 is inserted as the right child of 20.



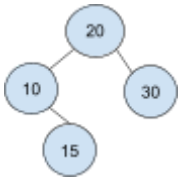
- The tree becomes unbalanced at node 10, as the balance factor of 10 is now -2 (right-heavy).
- **Double Rotation** (Left-Right Rotation) is needed:
    1. **Left Rotation** on node 20:
        - 30 becomes the new right child of 20.
        - 20 becomes the left child of 30.
    2. **Right Rotation** on node 10:
        - 20 becomes the new root of the tree.
        - 10 becomes the left child of 20.

After the rotations:



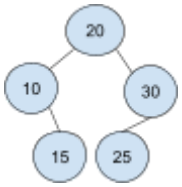
4. Insert 15:

- 15 is greater than 10 but less than 20, so it goes to the left of 20 and right of 10.



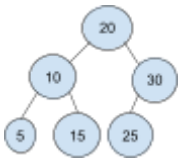
5. Insert 25:

- 25 is greater than 20 but less than 30, so it goes to the left of 30.



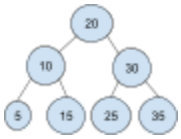
6. Insert 5:

- 5 is less than 10, so it is inserted as the left child of 10.



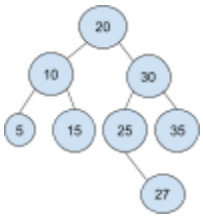
7. Insert 35:

- 35 is greater than 20, 30, and inserted as the right child of 30.



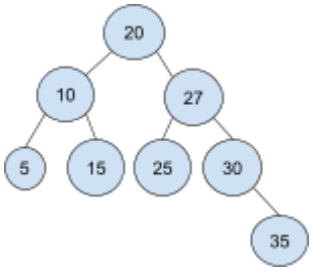
8. Insert 27:

- 27 is greater than 20, less than 30, and greater than 25, so it is inserted as the right child of 25.



- The tree becomes unbalanced at node 30 because the balance factor of 30 is +2 (left-heavy).
- **Double Rotation** (Right-Left Rotation) is needed:
    1. **Right Rotation** on node 25:
        - 27 becomes the new right child of 25.
        - 25 becomes the left child of 27.
    2. **Left Rotation** on node 30:
        - 27 becomes the new left child of 30.
        - 30 becomes the right child of 27.

After the rotations:



This is the final avl tree