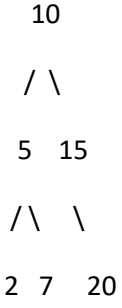
**CS23302 - Data Structures**

**Lab -5**

**Date: 07.08.25**

## Observation Questions

1. **Difference between Binary Tree and Binary Search Tree**
   **Question:** Given the following binary tree structure:




**2. In-order Traversal**
*Question:* For the above binary search tree, what is the output of in-order traversal?

**3. Time Complexity of BST Operations**
*Question:* If a binary search tree is balanced, what is the time complexity of search, insert, and delete operations? What if the tree is skewed?

4. Given a binary tree:


*What are the outputs of pre-order, in-order, and post-order traversals?*

**5. What are the Disadvantages of Unbalanced BST?**

## Execution Questions

1. **In-order Traversal Implementation**
   *Input:* A binary tree with nodes inserted as: `10`, `5`, `15`, `3`, `7`
   Write code to perform in-order traversal.

2. **BST Insertion Demonstration**
   *Input:* Starting with an empty BST, insert elements in order: `50`, `30`, `70`, `20`, `40`, `60`, `80`. Show the structure of BST after insertion.
3. **BST Deletion Scenario**
   *Input:* Given BST with nodes `50`, `30`, `70`, `20`, `40`, `60`, `80`. Demonstrate deletion of nodes with different cases:

- Leaf node (`20`)
- Node with one child (`30`)
- Node with two children (`70`)

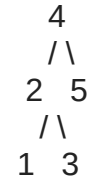
**4. Pre-order and Post-order Traversal Implementation**
*Input:* Using the BST from above, perform pre-order and post-order traversals.
*Expected Output:*

- Pre-order: `50 30 20 40 70 60 80`
- Post-order: `20 40 30 60 80 70 50`

**5. Handling Duplicate Values in BST**
*Input:* Insert values `25`, `15`, `25`, `10`, `20`, `25` into an empty BST.
*Task:* Show how duplicates are handled (e.g., ignoring, counting, or allowing duplicates on a specific side).

## Spot Questions

1. Convert a given Binary Search Tree (BST) into a sorted doubly linked list in-place. The left pointer should act as the previous pointer, and the right pointer as the next pointer.




   Output 1 <-> 2 <-> 3 <-> 4 <-> 5

2. Given a binary tree, find the size of the largest subtree that is a valid Binary Search Tree (BST).