Count Non-Leaf Nodes of Binary Tree GFG

Given a Binary Tree of size **N**, that should return the count of all the non-leaf nodes of the given binary tree.

Example:

```
          13
        /    \
      15      1
    /   \       \
  7       8       9
 /       / \        \
11    16   17       91
                   /
                 10
```

Output: The Non-Leaves Count: 7

Approach 1: Function to count non-leaf nodes recursively

- The **countNonLeavesRecursively** function counts the non-leaf nodes in the binary tree recursively.
- It initializes a count variable to 0 and then calls the helper function **inorderCount**.
- The **inorderCount** function performs an inorder traversal of the tree.
- In the traversal, if a node has at least one child (left or right), it increments the count of non-leaf nodes.
- Finally, it returns the count of non-leaf nodes.

Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.

Space Complexity: O(H), where H is the height of the binary tree due to the function call stack.

Approach 2: Function to count non-leaf nodes iteratively

- The **countNonLeavesIteratively** function counts the non-leaf nodes in the binary tree iteratively using a stack.

- It initializes a stack with the root node and a count variable to 0.
- In each iteration, it pops a node from the stack and checks if it has at least one child (left or right). If it does, it increments the count of non-leaf nodes.
- It then pushes the left and right children onto the stack for further processing.
- The loop continues until the stack is empty.
- Finally, it returns the count of non-leaf nodes.

Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.

Space Complexity: O(W), where W is the maximum width of the binary tree at any level due to the stack.