

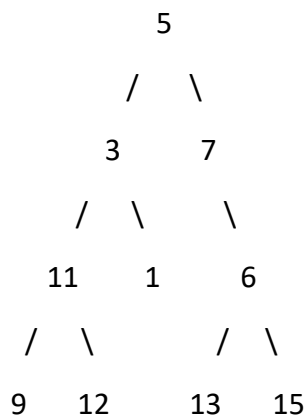
Find the Kth Ancestor of a Node in Binary Tree [GFG](#)

Given a binary tree of size **N**, a **node**, and a positive integer **k**., the function should return the **kth** ancestor of the given node in the binary tree. If there does not exist any such ancestor then return -1.

Note:

1. It is guaranteed that the **node** exists in the tree.
2. All the nodes of the tree have distinct values.

Example:



Target Node: 1, K = 2

Output: The 2nd Ancestor of 1 is: 5

Approach 1: Function to find the kth ancestor of a node in a binary tree

- Define a helper function **solve** that takes a binary tree node, a reference to **k** (remaining steps to the ancestor), and the **node** to find.
- In the **solve** function:
 - Base case: If the node is null, return null as no ancestor can be found.
 - If the current node's value matches the target **node**, it's considered an ancestor.
 - Recursively search for the **node** in the left and right subtrees.
 - Check if the target **node** was found in either the left or right subtree.
 - If the **k** value is positive, decrement it and check if it reached 0. If so, return the current node as the **k**-th ancestor.
- In the **kthAncestor** function:
 - Call the **solve** function to find the **k**-th ancestor of the target **node**.

- Check if the result is null or if it's the same as the target **node**.
- Return the data (value) of the **k**-th ancestor found.
- **Time Complexity: $O(N)$ as it visits each node exactly once.**
- **Space Complexity: $O(H)$ where H is the height of the tree (stack space for recursion).**