1. Implement Following Functions in the BinaryTree class-
    a. BinaryTree(const vector<T> & pre`, const vector<T> &in) – Constructor which creates tree from postorder and inorder.
    b. void iterativePostOrderTraversal()const;
    c. int diameter() const; - The diameter of a tree (sometimes called the width) is the number of nodes on the longest path between two leaves in the tree.
    d. void deleteAllHalfNodes() – Half nodes are those nodes which only have one child.
    e. void deleteAllLeafNodes()
    f. bool operator==(const BinaryTree &) const;– Return true if two trees are structurally identical.
    g. bool isBalanced() const; – i.e. depth of the left and right subtrees of every node differ by 1 or less.
    h. void printAllNodesWithoutSiblings() const; - Print all nodes which don't have a sibling.
    i. bool isThereAPathSum(const T & sum) const; – if the tree has a root-to-leaf path such that adding up all the values along the path equals the given number.
    j. void convertMirror() – mirror the current tree.
    k. vector<T> & maxRootToLeafPath() const – return root to leaf path which has maximum sum.
    l. Bool checkIfAllLeafNodesAreAtSameLevel() const – return true if all leaf nodes are the at same level.
    m. Bool checkIfTwoNodesWithSum(const T & sum) const – return true if there exist two nodes whose sum is equal to given value.
    n. const Node<T> * findLowestCommonAncestor(const T &el1, const T & el2) const;
    o. const Node<T> * findJustLargerThanK(const T & K)const; – Return address of the node just higher to K.
    p. void printZigZagLevelOrder() const;- Print the zig zag order i.e print level 1 from left to right, level 2 from right to left and so on. This means odd levels should get printed from left to right and even levels should be printed from right to left. Each level should be printed at a new line.

q. void prettyPrint() – Pretty print the binary tree [Hint find height of the tree]