class Solution {

public boolean isSymmetric(TreeNode root) {

if(root==null)return true;

//There is a need of other base cases on which we can perform our recursion.

//We couldn't do it with a single root because of

// 1) Value checking for symmtery.

// 2) Base case of recursion.

// 3)Comparing it with bpth the left sub tree and right subtree.

return IsMirror(root.left,root.right);

}

public boolean IsMirror(TreeNode root1,TreeNode root2)

{

if(root1==null && root2==null)return true;

if(root1==null || root2==null)return false;

return root1.val==root2.val && IsMirror(root1.left,root2.right)&& IsMirror(root1.right,root2.left);

}

}

Iterative implementation:

LinkedList<TreeNode> qleft = new LinkedList(),qright=new LinkedList();

if(root==null) return true;

qleft.add(root.left);

qright.add(root.right);

while(!qleft.isEmpty() && !qright.isEmpty()){

TreeNode currleft=qleft.remove();

TreeNode currright=qright.remove();

if(currleft==null && currright==null) continue;

if(currleft==null || currright==null) return false;

if(currleft.val!=currright.val) return false;

qleft.add(currleft.left);

qleft.add(currleft.right);

qright.add(currright.right);

qright.add(currright.left);

}

return true;

}