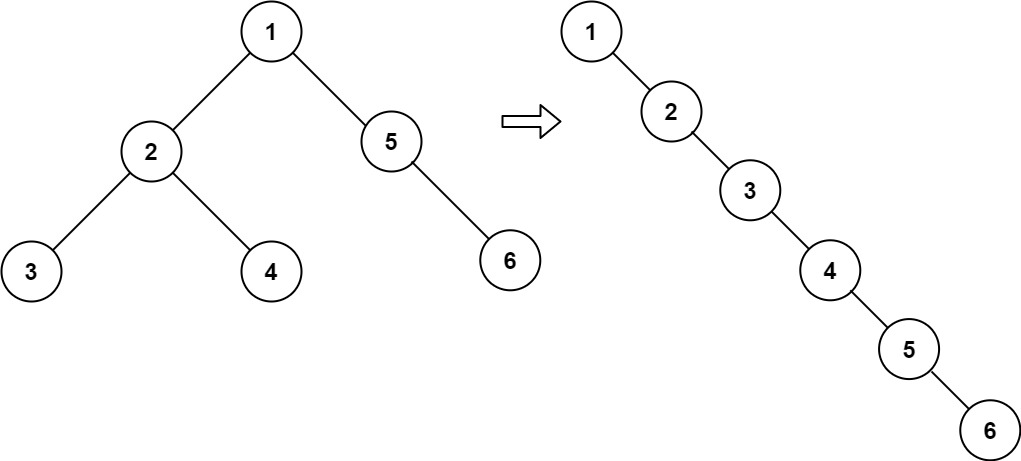
Given the root of a binary tree, flatten the tree into a "linked list":

* The "linked list" should use the same TreeNode class where the right child pointer points to the next node in the list and the left child pointer is always null.
* The "linked list" should be in the same order as a [**pre-order traversal**](https://en.wikipedia.org/wiki/Tree_traversal#Pre-order,_NLR) of the binary tree.

**Example 1:**



**Input:** root = [1,2,5,3,4,null,6]

**Output:** [1,null,2,null,3,null,4,null,5,null,6]

**Example 2:**

**Input:** root = []

**Output:** []

**Example 3:**

**Input:** root = [0]

**Output:** [0]

Solution:

|  |
| --- |
| // TC - O(N) |
|  | // SC - O(N) |
|  | class Solution { |
|  | TreeNode prev = null; |
|  | public void flatten(TreeNode root) { |
|  | if(root == null) return; |
|  |  |
|  | flatten(root.right); |
|  | flatten(root.left); |
|  |  |
|  | root.right = prev; |
|  | root.left = null; |
|  | prev = root; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | // TC - O(N) |
|  | // SC - O(1) |
|  | class Solution { |
|  | public void flatten(TreeNode root) { |
|  | TreeNode cur = root; |
|  | while (cur != null) |
|  | { |
|  | if(cur.left != null) |
|  | { |
|  | TreeNode pre = cur.left; |
|  | while(pre.right != null) |
|  | { |
|  | pre = pre.right; |
|  | } |
|  | pre.right = cur.right; |
|  | cur.right = cur.left; |
|  | cur.left = null; |
|  | } |
|  | cur = cur.right; |
|  | } |
|  | } |
|  | } |