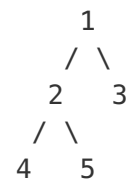
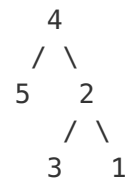
Binary Tree Upside Down

Given a binary tree where all the right nodes are either leaf nodes with a sibling (a left node that shares the same parent node) or empty, flip it upside down and turn it into a tree where the original right nodes turned into left leaf nodes. Return the new root.

For example:
Given a binary tree `{1,2,3,4,5}`,



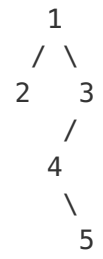
return the root of the binary tree `[4,5,2,#,#,3,1]`.



confused what `"{1,#,2,3}"` means? > read more on how binary tree is serialized on OJ.

OJ's Binary Tree Serialization:
The serialization of a binary tree follows a level order traversal, where '#' signifies a path terminator where no node exists below.

Here's an example:



The above binary tree is serialized as `"{1,2,3,#,#,4,#,#,5}"`.

Solution 1

```
public class Solution {
    public TreeNode UpsideDownBinaryTree(TreeNode root) {
        TreeNode curr = root;
        TreeNode prev = null;
        TreeNode next = null;
        TreeNode temp = null;

        while (curr != null) {
            next = curr.left;
            curr.left = temp;
            temp = curr.right;
            curr.right = prev;
            prev = curr;
            curr = next;
        }

        return prev;
    }
}

Just think about how you can save the tree information
you need before changing the tree structure.
```

written by Yida original link here

Solution 2

```
public TreeNode upsideDownBinaryTree(TreeNode root) {
  if (root == null || root.left == null && root.right == null)
    return root;

  TreeNode newRoot = upsideDownBinaryTree(root.left);

  root.left.left = root.right;
  root.left.right = root;

  root.left = null;
  root.right = null;

  return newRoot;
}
```

written by jeantimex original link here

Solution 3

```
TreeNode* upsideDownBinaryTree(TreeNode* root) {
    if (!root || !root->left) return root;
    TreeNode* cur_left = root->left;
    TreeNode* cur_right = root->right;
    TreeNode* new_root = upsideDownBinaryTree(root->left);
    cur_left->right = root;
    cur_left->left = cur_right;
    root->left = nullptr;
    root->right = nullptr;
    return new_root;
}
```

written by lchen77 original link here