

```
1  /* Given a binary tree, flatten it to a linked list in-place.
```

```
2
```

```
3  For example, given the following tree:
```

```
4
```

```
5      1
```

```
6     /\
```

```
7    2  5
```

```
8   /\  \
```

```
9  3 4  6
```

```
10 The flattened tree should look like:
```

```
11
```

```
12  1
```

```
13   \
```

```
14  2
```

```
15   \
```

```
16  3
```

```
17   \
```

```
18  4
```

```
19   \
```

```
20  5
```

```
21   \
```

```
22  6 */
```

```
23
```

```
24
```

```
25 class TreeNode {
```

```
26     int val;
```

```
27     TreeNode left;
```

```
28     TreeNode right;
```

```
29
```

```
30     TreeNode(int x) {
```

```
31         val = x;
```

```
32     }
```

```
33 }
```

```
34
```

```
35
```

```
36 class Solution {
```

```
37     public void flatten(TreeNode root) {
```

```
38
```

```
39         flattenHelper(root);
```

```
40     }
```

```
41
```

```
42     private TreeNode flattenHelper(TreeNode root) {
```

```
43
```

```
44         if (root == null) {
```

```
45             return null;
```

```
46         }
```

```
47
```

```
48         TreeNode leftLeaf = root.left;
```

```
49         TreeNode rightLeaf = root.right;
```

```
50
```

```
51         root.left = null;
```

```
52         root.right = flattenHelper(leftLeaf);
```

```
53
```

```
54         TreeNode temp = root;
```

```
55         while (temp.right != null) {
```

```
56             temp = temp.right;
```

```
57         }
```

```
58
```

```
59         temp.right = flattenHelper(rightLeaf);
```

```
60
```

```
61         return root;
```

```
62     }
```

```
63 }
```