



CS 106B section 7 (spring 2026 🌱)

Give 'Em the Axe

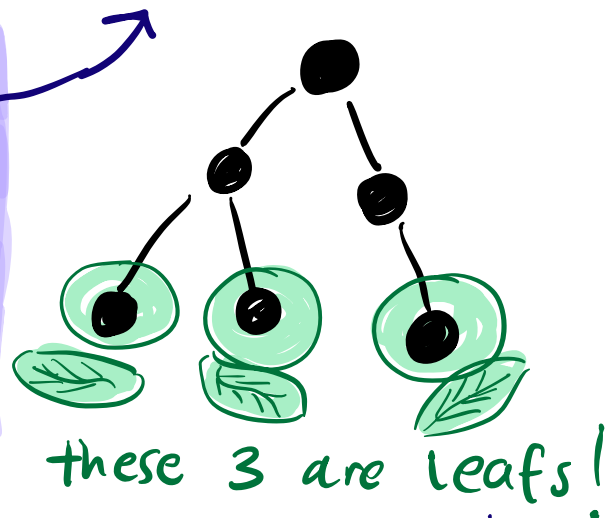
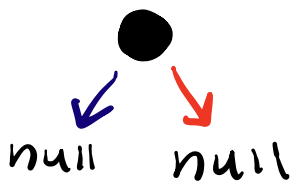
Ben Yan[★]_★

Write a function

```
void removeLeaves(TreeNode* & root)
```

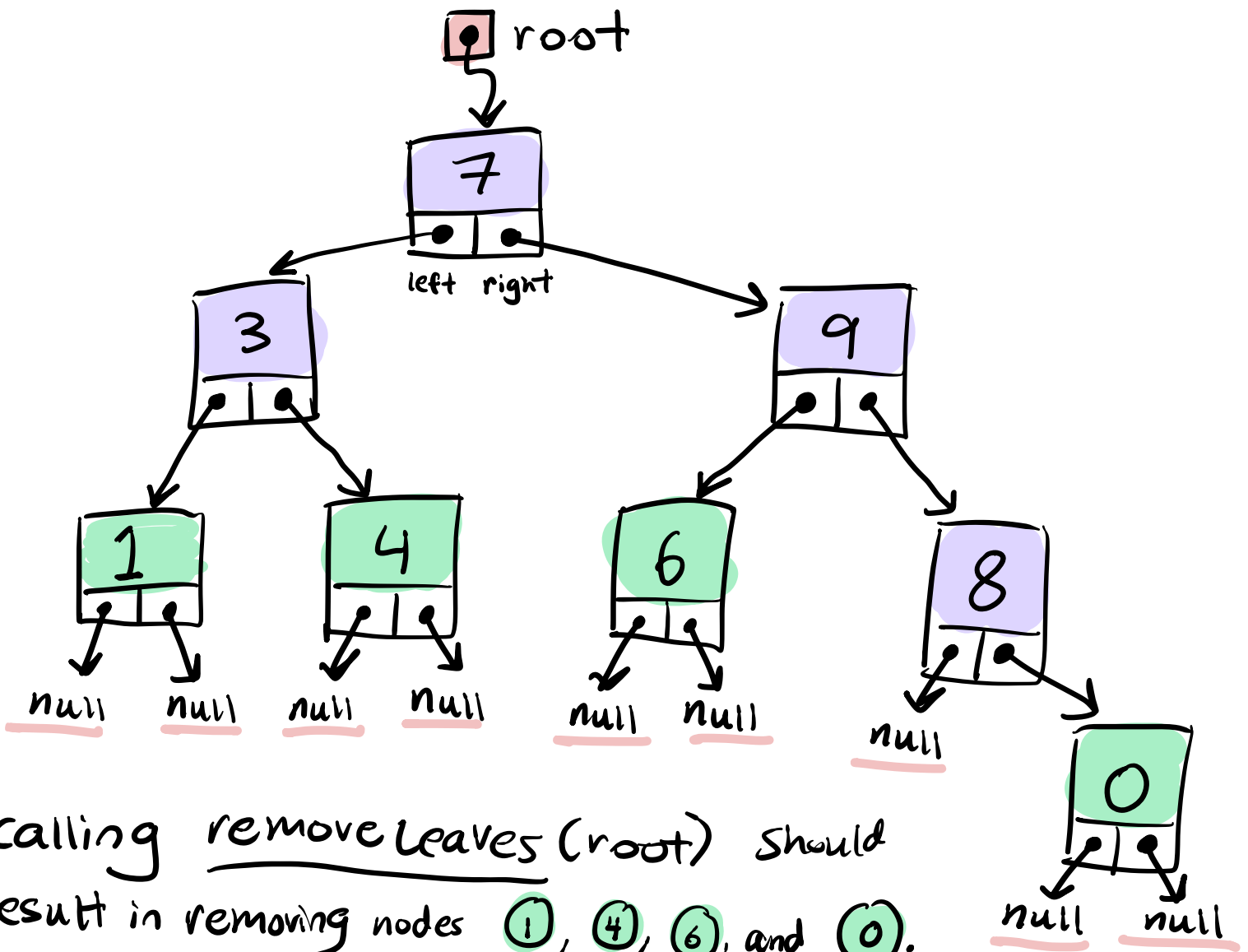
that accepts (by reference) a pointer to the
TreeNode that is the root of some tree
or subtree, and removes all leaves.

A leaf is a `TreeNode`
without children.

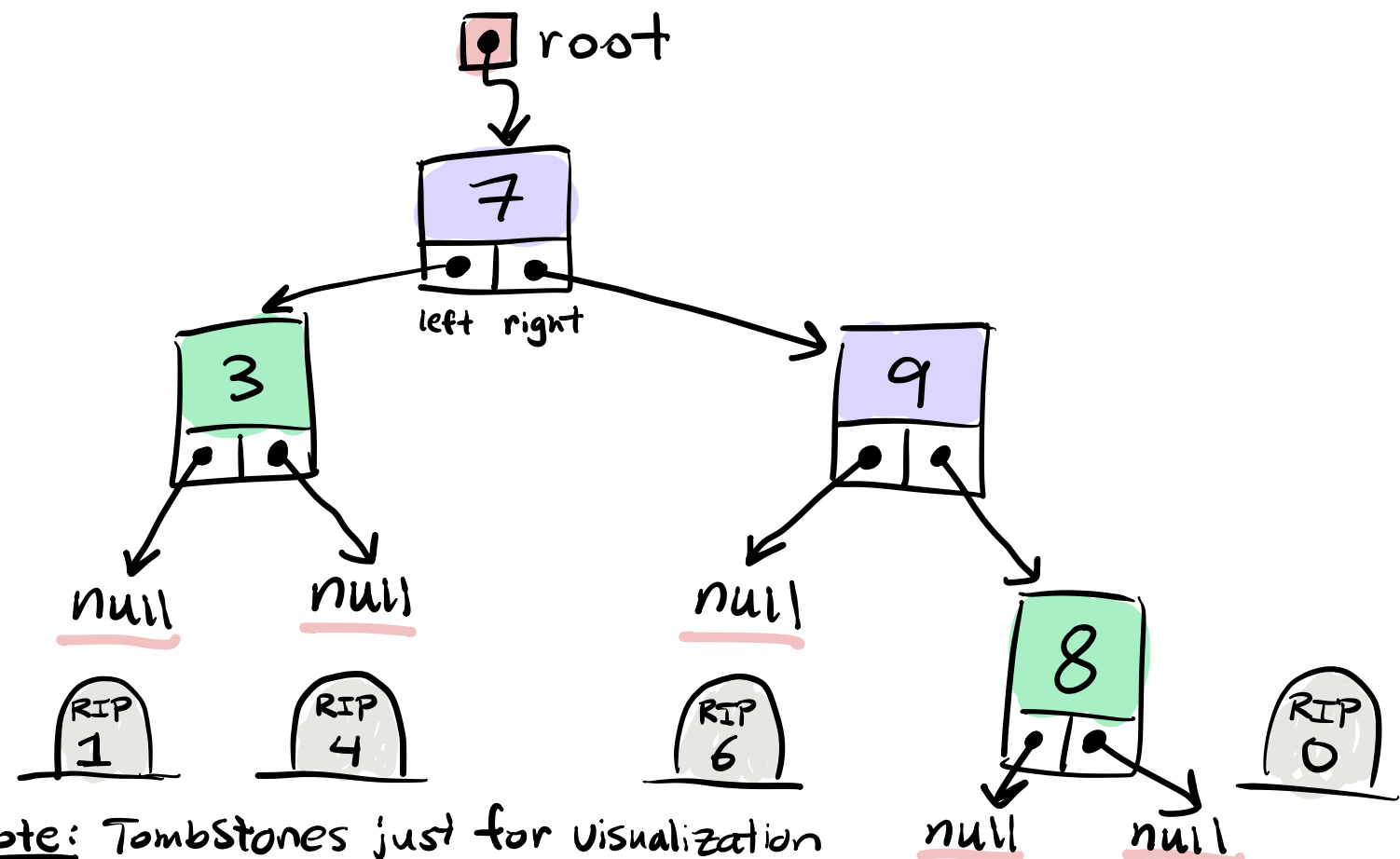


Equivalently, both the left and right fields of the struct are null ptr.

As an example, consider the tree below.

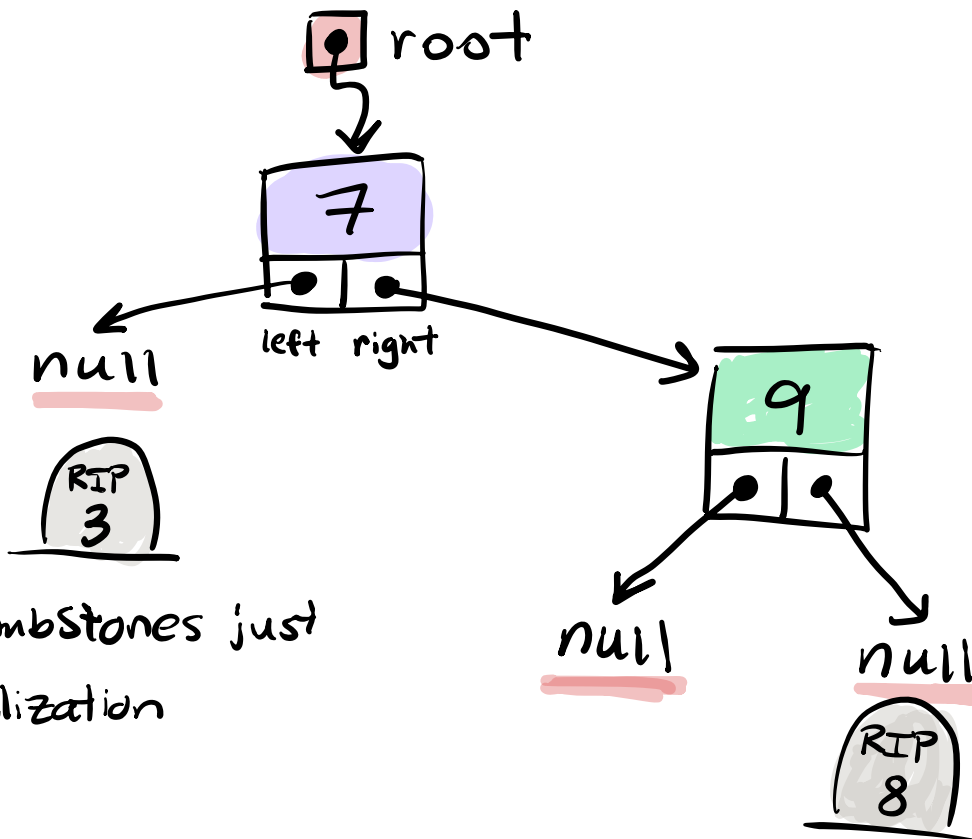


calling removeLeaves(root) should result in removing nodes 1, 4, 6, and 0.



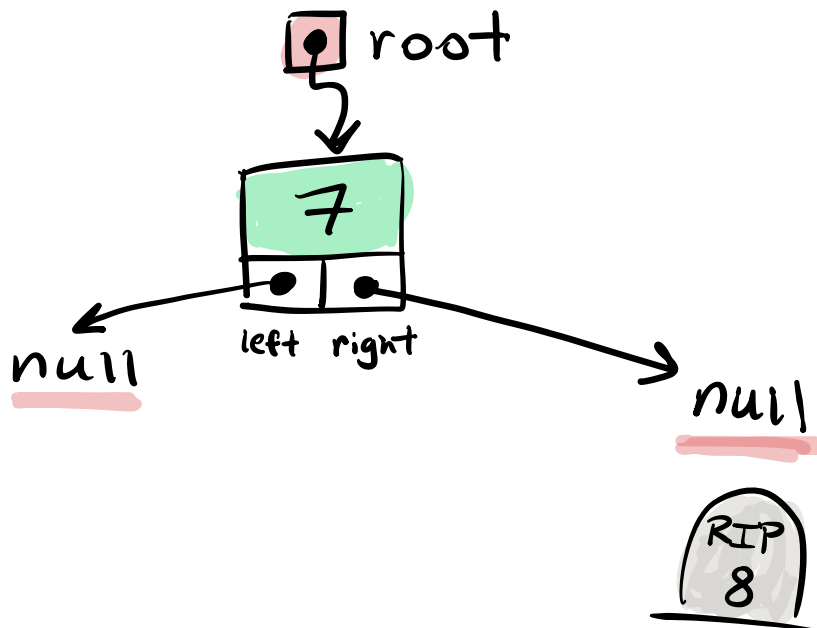
Note: Tombstones just for visualization

A second call removeLeaves(root) would then remove the (now) leaves ③, ⑧, resulting in

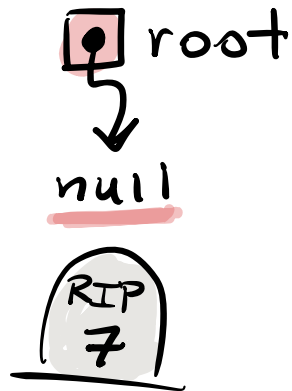


Note: TombStones just for visualization

Now, the (solitary) leaf is ⑨, so a 3rd call to removeLeaves(root) would delete that.



Finally, the root node itself is a leaf 🍃 (ie. no children), so calling removeLeaves(root) would delete that, and set the root pointer (passed in by reference) to nullptr, signifying an empty tree.



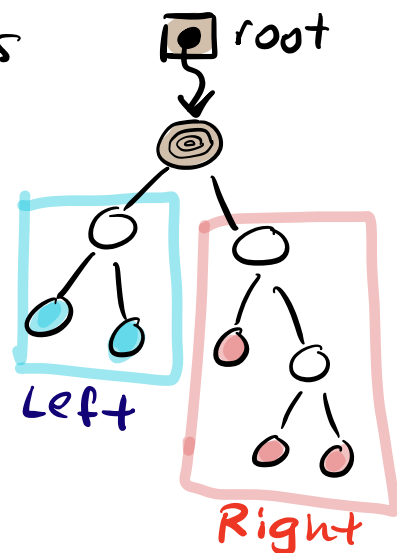
Calling removeLeaves(root) on an empty tree should do nothing, and our last requirement is to free any memory for removed (axed) nodes.

? How do we do this?

Think recursively! To remove the leaves on a tree, we need to (generally) remove

- 🔪 (1) Leaves on the left subtree
- 🔪 (2) Leaves on the right subtree

Visually, think of dropping two axes:
one falling on the left, the other on right.



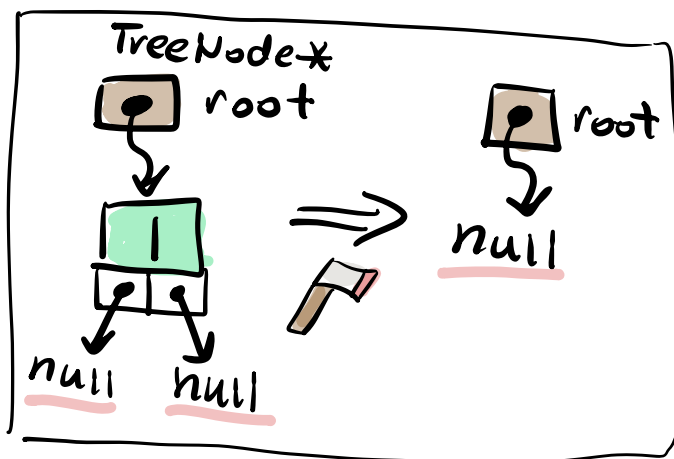
Base Case(s):

* Tree is empty (ie. root is nullptr )

↳ Do nothing! That is, just return!

* Tree is just the root node
(ie. root is a leaf )

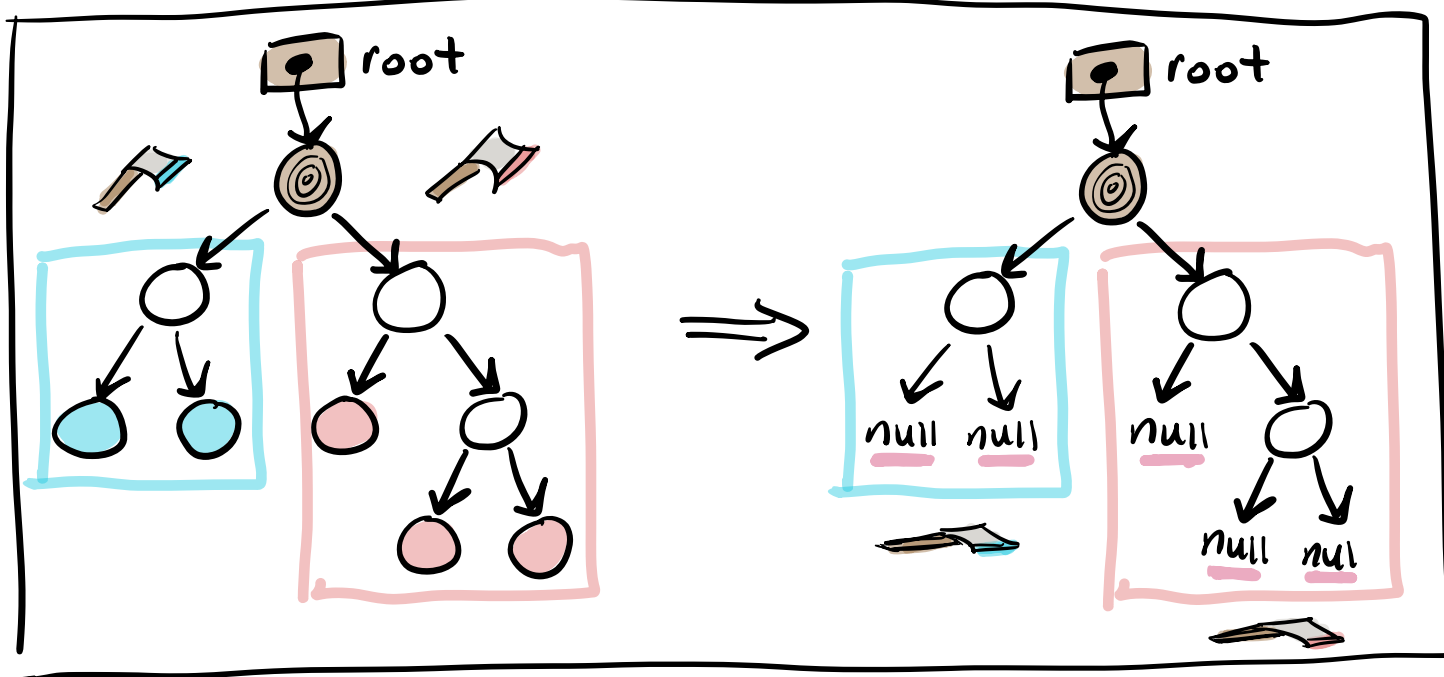
↳ delete[] the root node
and set the root pointer to nullptr.



Recursive Case

* Tree is not empty, and not just the root node.

↳ Call removeLeaves() on both the left and right subtrees of the input root node.



A C++ implementation is presented below :D

```
void removeLeaves(TreeNode*& root){
```

```
    /* base case(s) */
```

```
    if (root == nullptr){
```

```
        return; // do nothing if empty tree
    }
```

```
    if (root->left == nullptr
```

```
        && root->right == nullptr ) {
```

```
        // remove root if leaf, and set to null
        delete root;
```

```
        root = nullptr;
```

```
    }
```

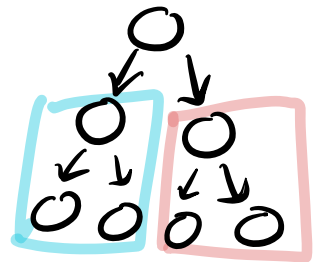
```
    /* recursive case */
```

```
    // otherwise, drop axes on both the left +
```

```
    // and right subtrees to take down their leaves
```

```
    removeLeaves(root->left);
```

```
    removeLeaves(root->right);
```



```
}
```