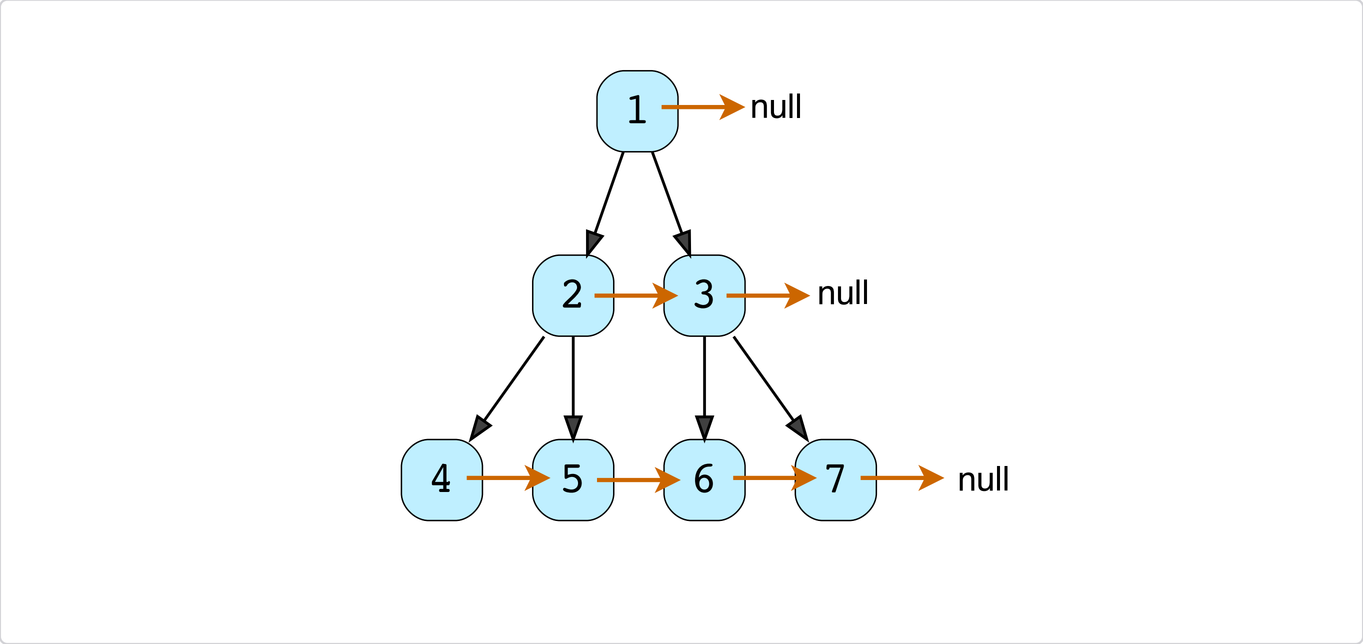
Connect Level Order Siblings (medium)

We'll cover the following

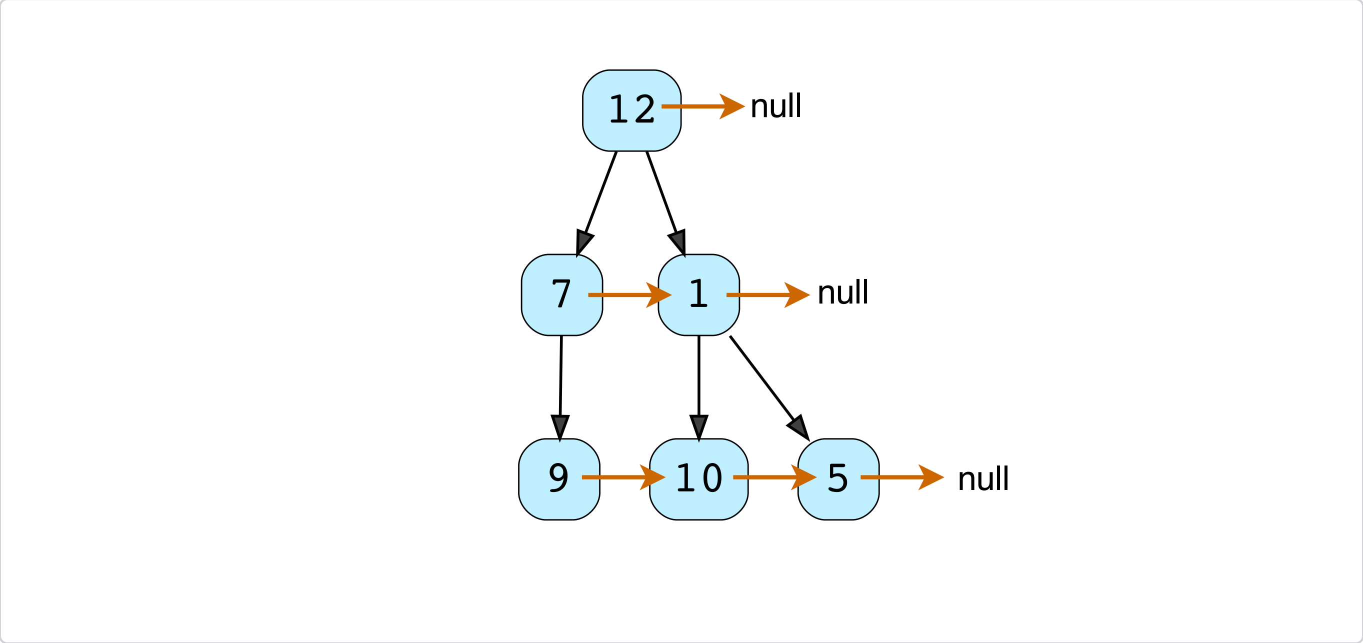
Problem Statement#

Given a binary tree, connect each node with its level order successor. The last node of each level should point to a `null` node.

Example 1:

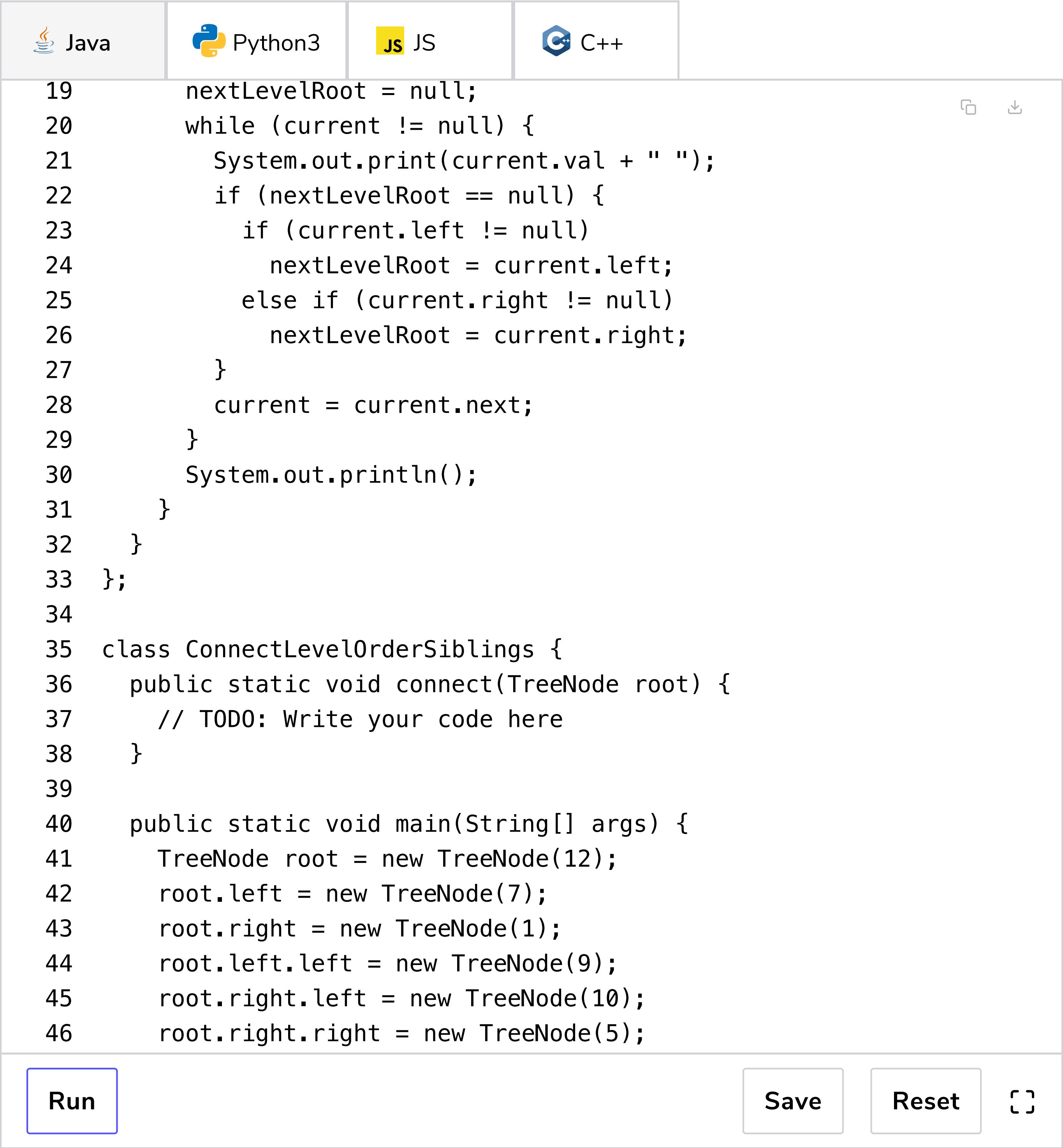


Example 2:



Try it yourself#

Try solving this question here:



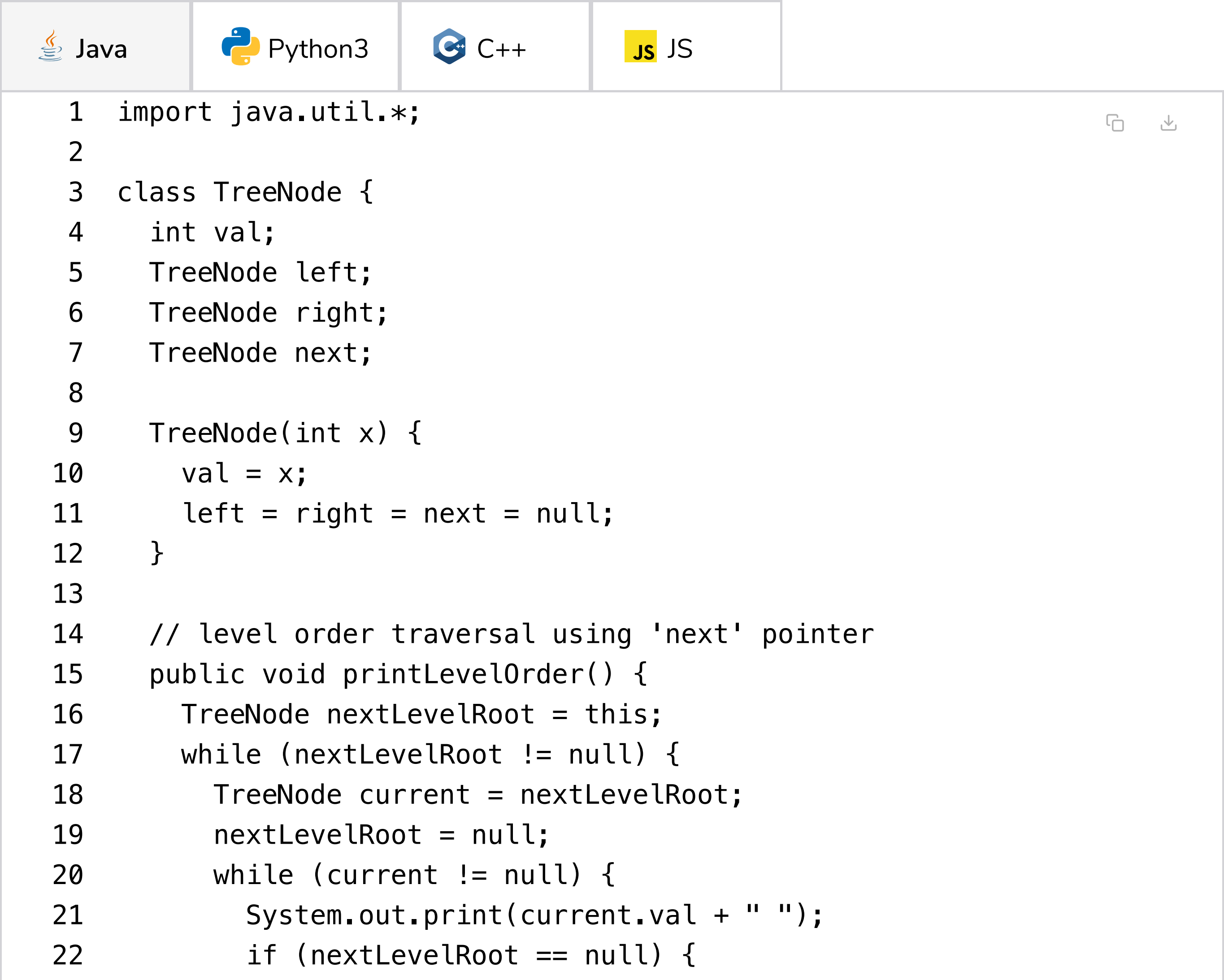
```java
      nextLevelRoot = null;
      while (current != null) {
        System.out.print(current.val + " ");
        if (nextLevelRoot == null) {
          if (current.left != null)
            nextLevelRoot = current.left;
          else if (current.right != null)
            nextLevelRoot = current.right;
        }
        current = current.next;
      }
      System.out.println();
    }
  }
};

class ConnectLevelOrderSiblings {
  public static void connect(TreeNode root) {
    // TODO: Write your code here
  }

  public static void main(String[] args) {
    TreeNode root = new TreeNode(12);
    root.left = new TreeNode(7);
    root.right = new TreeNode(1);
    root.left.left = new TreeNode(9);
    root.right.left = new TreeNode(10);
    root.right.right = new TreeNode(5);
```

Solution#

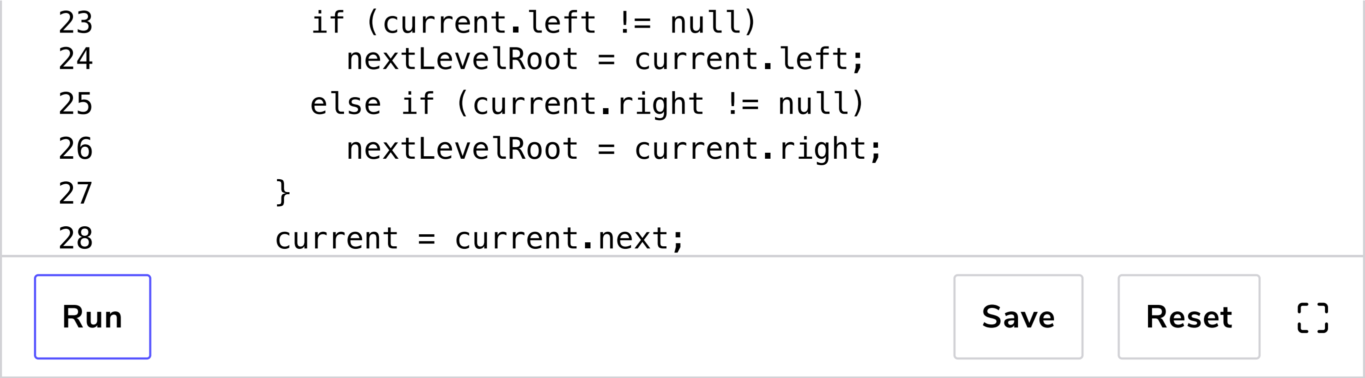
This problem follows the Binary Tree Level Order Traversal (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5726607939469312/) pattern. We can follow the same **BFS** approach. The only difference is that while traversing a level we will remember the previous node to connect it with the current node.

Code#

Here is what our algorithm will look like; only the highlighted lines have changed:



```java
import java.util.*;

class TreeNode {
  int val;
  TreeNode left;
  TreeNode right;
  TreeNode next;

  TreeNode(int x) {
    val = x;
    left = right = next = null;
  }

  // level order traversal using 'next' pointer
  public void printLevelOrder() {
    TreeNode nextLevelRoot = this;
    while (nextLevelRoot != null) {
      TreeNode current = nextLevelRoot;
      nextLevelRoot = null;
      while (current != null) {
        System.out.print(current.val + " ");
        if (nextLevelRoot == null) {
```



```
          if (current.left != null)
            nextLevelRoot = current.left;
          else if (current.right != null)
            nextLevelRoot = current.right;
        }
        current = current.next;
```

Time complexity#

The time complexity of the above algorithm is $O(N)$, where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

Space complexity#

The space complexity of the above algorithm will be $O(N)$, which is required for the queue. Since we can have a maximum of $N/2$ nodes at any level (this could happen only at the lowest level), therefore we will need $O(N)$ space to store them in the queue.

← Back

Next →

Level Order Successor (easy)

Problem Challenge 1

☑ Mark as Completed

Report an Issue