Solution 1

```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class BinaryTree {
  constructor(value) {
    this.value = value;
    this.left = null;
    this.right = null;
  }
}

// O(n) time | O(d) space - where n is the number of nodes in the Binary Tree and d is the depth (height) of the Binary Tree
function rightSiblingTree(root) {
  mutate(root, null, null);
  return root;
}

function mutate(node, parent, isLeftChild) {
  if (node === null) return;
  const {left, right} = node;
  mutate(left, node, true);
  if (parent === null) {
    node.right = null;
  } else if (isLeftChild) {
    node.right = parent.right;
  } else {
    if (parent.right === null) {
      node.right = null;
    } else {
      node.right = parent.right.left;
    }
  }
  mutate(right, node, false);
}

exports.BinaryTree = BinaryTree;
exports.rightSiblingTree = rightSiblingTree;

```