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

#include <vector>
using namespace std;

class BinaryTree {
public:
  int value;
  BinaryTree *left = NULL;
  BinaryTree *right = NULL;

  BinaryTree(int value);
};

vector<BinaryTree *> flattenTree(BinaryTree *node);
void connectNodes(BinaryTree *one, BinaryTree *two);
BinaryTree *getLeftMost(BinaryTree *node);

// 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
BinaryTree *flattenBinaryTree(BinaryTree *root) {
  flattenTree(root);
  return getLeftMost(root);
}

vector<BinaryTree *> flattenTree(BinaryTree *node) {
  BinaryTree *leftMost;
  BinaryTree *rightMost;

  if (node->left == NULL) {
    leftMost = node;
  } else {
    vector<BinaryTree *> leftAndRightMostNodes = flattenTree(node->left);
    connectNodes(leftAndRightMostNodes[1], node);
    leftMost = leftAndRightMostNodes[0];
  }

  if (node->right == NULL) {
    rightMost = node;
  } else {
    vector<BinaryTree *> leftAndRightMostNodes = flattenTree(node->right);
    connectNodes(node, leftAndRightMostNodes[0]);
    rightMost = leftAndRightMostNodes[1];
  }

  return {leftMost, rightMost};
}

void connectNodes(BinaryTree *left, BinaryTree *right) {
  left->right = right;
  right->left = left;
}

BinaryTree *getLeftMost(BinaryTree *node) {
  while (node->left != NULL) {
    node = node->left;
  }
  return node;
}

```