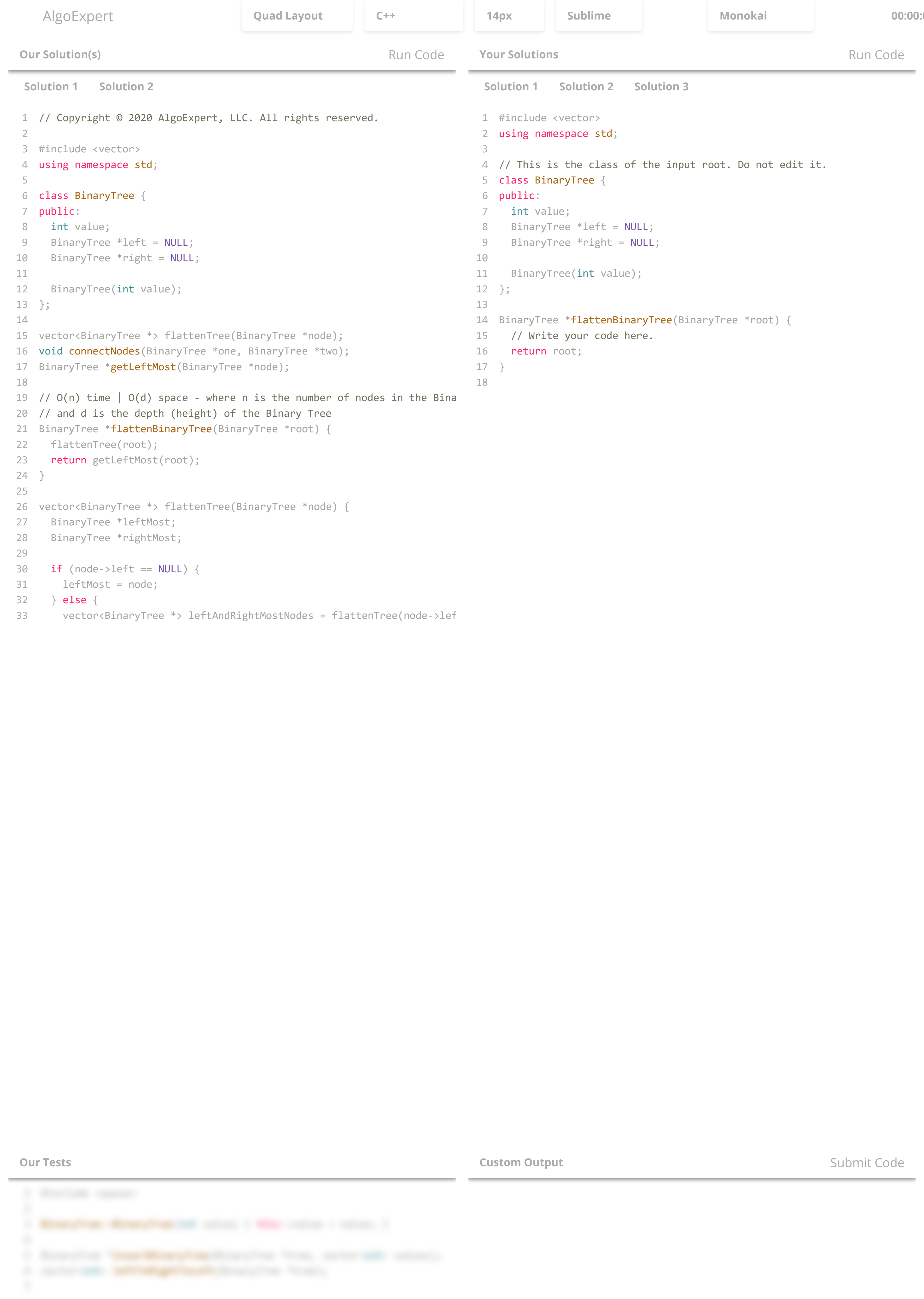

Our Solution(s)

```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 Bina
// 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->lef
```

Your Solutions

```cpp
#include <vector>
using namespace std;

// This is the class of the input root. Do not edit it.
class BinaryTree {
public:
  int value;
  BinaryTree *left = NULL;
  BinaryTree *right = NULL;

  BinaryTree(int value);
};

BinaryTree *flattenBinaryTree(BinaryTree *root) {
  // Write your code here.
  return root;
}

```

Our Tests

```cpp
#include <queue>

BinaryTree::BinaryTree(int value) { this->value = value; }

BinaryTree *insertBinaryTree(BinaryTree *tree, vector<int> values);
vector<int> leftToRightToLeft(BinaryTree *tree);

```

Custom Output

Run or submit code when you're ready.