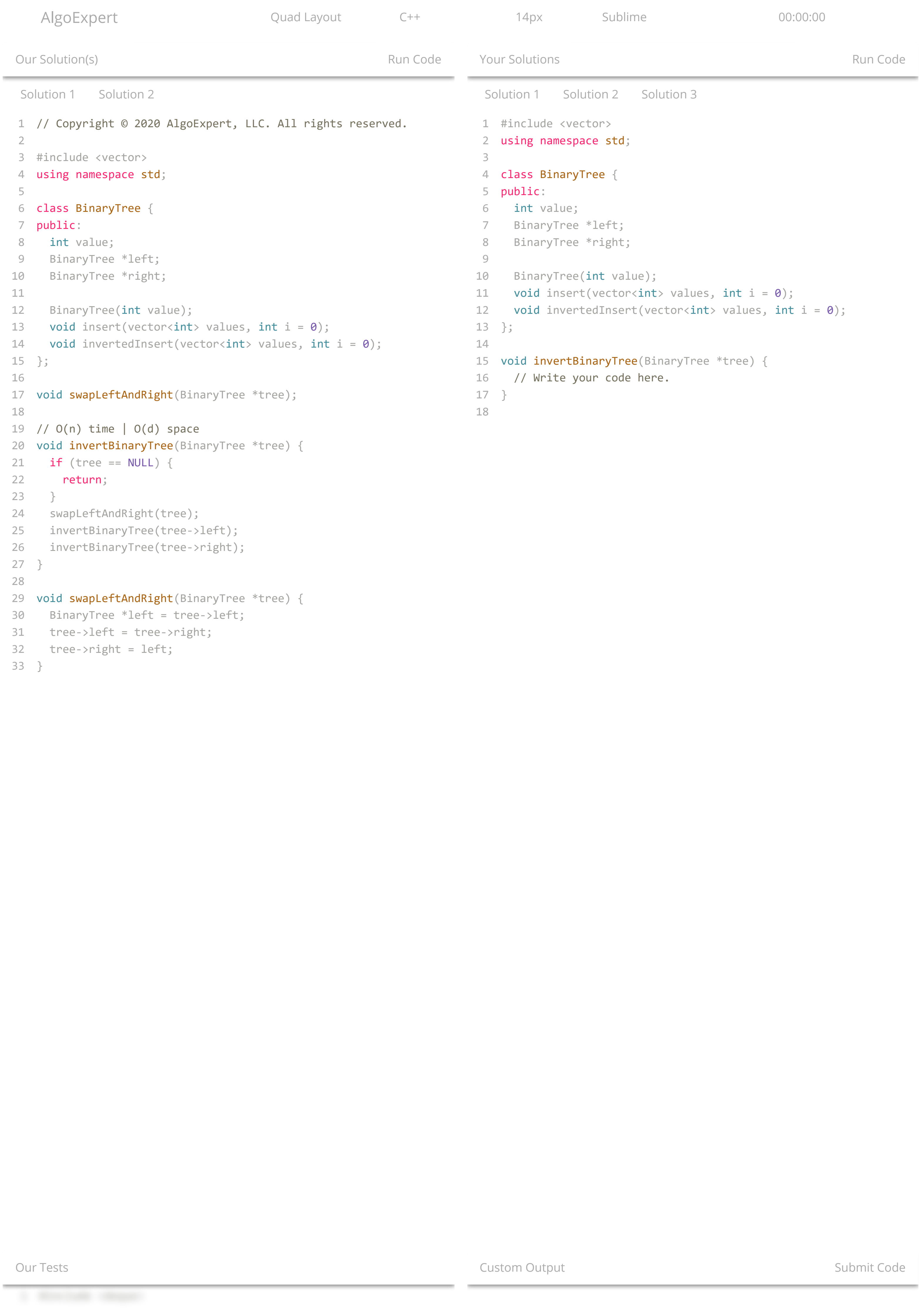

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

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

  BinaryTree(int value);
  void insert(vector<int> values, int i = 0);
  void invertedInsert(vector<int> values, int i = 0);
};

void swapLeftAndRight(BinaryTree *tree);

// O(n) time | O(d) space
void invertBinaryTree(BinaryTree *tree) {
  if (tree == NULL) {
    return;
  }
  swapLeftAndRight(tree);
  invertBinaryTree(tree->left);
  invertBinaryTree(tree->right);
}

void swapLeftAndRight(BinaryTree *tree) {
  BinaryTree *left = tree->left;
  tree->left = tree->right;
  tree->right = left;
}
```

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

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

  BinaryTree(int value);
  void insert(vector<int> values, int i = 0);
  void invertedInsert(vector<int> values, int i = 0);
};

void invertBinaryTree(BinaryTree *tree) {
  // Write your code here.
}
```

```

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

void BinaryTree::insert(vector<int> values, int i) {
  if (i >= values.size()) {
    return;
  }
  deque<BinaryTree *> queue;
  queue.push_back(this);
  while (queue.size() > 0) {
    BinaryTree *current = queue.front();
    queue.pop_front();
    if (current->left == NULL) {
      current->left = new BinaryTree(values[i]);
      break;
    }
    queue.push_back(current->left);
```

Run or submit code when you're ready.