Our Solution(s) Run Code

Solution 1 Solution 2

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

#include <vector>
#include <deque>
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(n) space
void invertBinaryTree(BinaryTree *tree) {
  deque<BinaryTree *> queue;
  queue.push_back(tree);
  while (queue.size() > 0) {
    BinaryTree *current = queue.front();
    queue.pop_front();
    if (current == NULL) {
      continue;
    }
    swapLeftAndRight(current);
    queue.push_back(current->left);
    queue.push_back(current->right);
  }
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```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.
}

```

Our Tests

Custom Output Submit Code

```cpp

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

void BinaryTree::insertLevelOrder(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.