Our Solution(s) Run Code

Solution 1

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

#include <vector>
using namespace std;

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

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

// O(n) time | O(1) space
void iterativeInOrderTraversal(BinaryTree *tree,
                               void (*callback)(BinaryTree *tree)) {
  BinaryTree *previousNode = NULL;
  BinaryTree *currentNode = tree;
  while (currentNode != NULL) {
    BinaryTree *nextNode;
    if (previousNode == NULL || previousNode == currentNode->parent) {
      if (currentNode->left != NULL) {
        nextNode = currentNode->left;
      } else {
        (*callback)(currentNode);
        nextNode = currentNode->right != NULL ? currentNode->right
                                              : currentNode->parent;
      }
    } else if (previousNode == currentNode->left) {
      (*callback)(currentNode);
```

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 *parent;

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

void iterativeInOrderTraversal(BinaryTree *tree,
                               void (*callback)(BinaryTree *tree)) {
  // Write your code here.
}
```

Our Tests

Custom Output Submit Code

```
}

void BinaryTree::insert(vector<int> values, int i) {
  if (i >= values.size()) {
    return;
  }
  Node<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], current);
      break;
    }
```

**Run or submit code when you're ready.**