

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 # Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 # O(n) time | O(1) space
4 def iterativeInOrderTraversal(tree, callback):
5     previousNode = None
6     currentNode = tree
7     while currentNode is not None:
8         if previousNode is None or previousNode == currentNode.parent:
9             if currentNode.left is not None:
10                 nextNode = currentNode.left
11             else:
12                 callback(currentNode)
13                 nextNode = currentNode.right if currentNode.right is not None else None
14         elif previousNode == currentNode.left:
15             callback(currentNode)
16             nextNode = currentNode.right if currentNode.right is not None else None
17         else:
18             nextNode = currentNode.parent
19         previousNode = currentNode
20         currentNode = nextNode
21
```

Solution 1Solution 2Solution 3

```
1 def iterativeInOrderTraversal(tree, callback):
2     # Write your code here.
3     pass
4
```

Our Tests

Custom Output

Submit Code

```

14     left, right = None
15     left, right = None
16     left, right = None
17
18     # Base case: if the list is empty, return None
19     if not nums:
20         return None
21
22     # Recursive case: find the middle element and split the list
23     mid = len(nums) // 2
24     left = nums[:mid]
25     right = nums[mid:]
26
27     # Recursively sort the left and right halves
28     left = sort(left)
29     right = sort(right)
30
31     # Merge the sorted halves
32     return merge(left, right)
33
34 # Example usage
35 nums = [5, 2, 8, 1, 9, 4, 7, 3, 6, 0]
36 sorted_nums = sort(nums)
37 print(sorted_nums)

```

Run or submit code when you're ready.