100

Venkata Naga Sri Sai Pranavi Kolipaka

Other

PDF generated at: 21 Jul 2025 16:38:00 UTC

View this report on HackerRank

Score

100% • 80 / 80

scored in TIP102: Unit 8 Version A (Standard) - Summer 2025 in 60 min 4 sec on 21 Jul 2025 08:36:08 PDT

Candidate Information

Email	kolipakavnssaipranavi@gmail.com
Test	TIP102: Unit 8 Version A (Standard) - Summer 2025
Candidate Packet	View
Taken on	21 Jul 2025 08:36:08 PDT
Time taken	60 min 4 sec/ 90 min
Personal Member ID	129054
Email Address with CodePath	kolipakavnssaipranavi@gmail.com
Github username with CodePath	Pranavi2002
Invited by	CodePath

Skill Distribution

There is no associated skills data that can be shown for this assessment

Tags Distribution

There is no associated tags data that can be shown for this assessment

Questions

Coding Questions • 60 / 60

Status	No.	Question	Time Taken	Skill	Score	Code Quality
✓	1	Inorder Traversal of Binary Tree Coding	5 min 10 sec	-	20/20	-
✓	2	Binary Search Tree (BST) Insertion Coding	22 min 30 sec	-	20/20	-

✓	3	Root Equals Sum of Descendants Coding	22 min 6 sec	-	20/20	-

Multiple Choice + Debugging • 20 / 20

Status	No.	Question	Time Taken	Skill	Score	Code Quality
✓	4	Debug this code Coding	2 min 15 sec	-	5/5	-
✓	5	What is the time complexity of mystery_function()? Multiple Choice	4 min 52 sec	-	5/5	-
✓	6	Which of the following best represents the values of the tree? Multiple Choice	1 min 7 sec	-	5/5	-
✓	7	Which of the following arrays best represents its preorder traversal? Multiple Choice	1 min 45 sec	-	5/5	-

1. Inorder Traversal of Binary Tree

✓ Correct

Coding

Question description

Given the `root` of a binary tree, return a list of integers representing the inorder traversal of its nodes' values.

Example 1:
Input: root = [1,null,2,3]

Output: [1, 3, 2]

Example 2:
Input: root = []
Output: []

Example 3:
Input: root = [1]
Output: [1]

Candidate's Solution

Language used: Python 3

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right


def inorder_traversal(root):
    # Write your code here
    if not root:
```

```
        return []
    result = []
    result += inorder_traversal(root.left)
    result.append(root.val)
    result += inorder_traversal(root.right)

    return result
def build_tree_from_list(values):
    if not values:
        return None

    root = TreeNode(values[0])
    queue = [root]
    i = 1
    while i < len(values):
        current = queue.pop(0)
        if values[i] is not None:
            current.left = TreeNode(values[i])
            queue.append(current.left)
        i += 1
        if i < len(values) and values[i] is not None:
            current.right = TreeNode(values[i])
            queue.append(current.right)
        i += 1

    return root

if __name__ == '__main__':
    outfile = open(os.environ['OUTPUT_PATH'], 'w')
    input_data = sys.stdin.read().strip()

    input_data = input_data.splitlines()

    for data in input_data:
        if data.strip() == "":
            continue

        data = data.replace('null', 'None')
        tree_list = ast.literal_eval(data)

        root = build_tree_from_list(tree_list)
        result = inorder_traversal(root)
        outfile.write(str(result) + '\n')
    outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case	Easy	Hidden	Success	0	0.0303 sec	11 KB
Empty Tree	Easy	Hidden	Success	0	0.0308 sec	10.9 KB
Single Node Root	Easy	Hidden	Success	0	0.0322 sec	10.9 KB
Left-Skewed Tree	Easy	Hidden	Success	0	0.0432 sec	10.9 KB
Complete Binary Tree	Easy	Hidden	Success	0	0.0327 sec	10.8 KB
Tree with `None` nodes	Easy	Hidden	Success	0	0.0315 sec	10.9 KB
Tree with Single Child Nodes	Easy	Hidden	Success	0	0.0438 sec	10.9 KB
Large Tree	Easy	Hidden	Success	0	0.0297 sec	11 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0309 sec	11 KB

No comments.

2. Binary Search Tree (BST) Insertion

Correct

Coding

Question description

Given the `root` of a binary search tree and a `value`, insert a new node with `value` into the BST. Return the root of the BST after the insertion.

Example 1:
Input: root = [4,2,7,1,3], val = 5
Output: [4,2,7,1,3,5]

Example 2:
Input: root = [40,20,60,10,30,50,70], val = 25
Output: [40,20,60,10,30,50,70,None,None,25]

Candidate's Solution

Language used: Python 3

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right


def insert_into_bst(root, val):
    # Write your code here
    if not root:
```

```
21          return TreeNode(val)
22      if val < root.val:
23          root.left = insert_into_bst(root.left, val)
24      else:
25          root.right = insert_into_bst(root.right, val)
26      return root
27
28  def build_tree(values):
29      if not values:
30          return None
31      root = TreeNode(values[0])
32      for val in values[1:]:
33          insert_into_bst(root, val)
34      return root
35
36  def bst_to_list(root):
37      if not root:
38          return []
39      result = []
40      queue = [root]
41      while queue:
42          node = queue.pop(0)
43          if node:
44              result.append(node.val)
45              queue.append(node.left)
46              queue.append(node.right)
47          else:
48              result.append(None)
49      # Remove trailing 'None' values
50      while result and result[-1] is None:
51          result.pop()
52      return result
53
54  if __name__ == '__main__':
55      input_data = sys.stdin.read().strip().splitlines()
56
57      for data in input_data:
58          tree_data, val = data.split('],')
59          tree_data += ']'
60          val = int(val.strip())
61
62          tree_list = ast.literal_eval(tree_data)
63
64          root = build_tree(tree_list)
65          result = insert_into_bst(root, val)
66
```

```
67          print(bst_to_list(result))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Insertion Case	Easy	Hidden	Success	0	0.029 sec	11 KB
Insertion at the Root	Easy	Hidden	Success	0	0.0305 sec	10.9 KB
Insertion in a Single-Node Tree	Easy	Hidden	Success	0	0.0286 sec	10.9 KB
Insertion Causing Multiple Levels	Easy	Hidden	Success	0	0.0278 sec	10.8 KB
Insertion at a Leaf Node	Easy	Hidden	Success	0	0.031 sec	10.9 KB
Insertion into a Full Binary Tree	Easy	Hidden	Success	0	0.034 sec	10.9 KB
Insertion into a Tree with Duplicates	Easy	Hidden	Success	0	0.0327 sec	10.9 KB
Insertion of a Minimum Value	Easy	Hidden	Success	0	0.0345 sec	10.9 KB
Insertion of a Maximum Value	Easy	Hidden	Success	0	0.0324 sec	10.9 KB

Large Input Tree	Easy	Hidden	Success	0	0.0317 sec	10.9 KB
Very Large Value Insertion	Easy	Hidden	Success	0	0.0342 sec	10.9 KB
Very Small Value Insertion	Easy	Hidden	Success	0	0.0375 sec	10.9 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0324 sec	10.9 KB

No comments.

3. Root Equals Sum of Descendants

Correct

Coding

Question description

Given the `root` of a binary tree, write a function that returns `True` if the value of `root` is equal to the sum of the values of all its descendants. Return `False` otherwise.

Example 1:
Input: root = [10, 4, 6]
Output: True
Explanation: 10 = 4 + 6

Example 2:
Input: root = [5, 3, 1, 1, 1, 1, 1]
Output: False
Explanation: 5 != 3 + 1 + 1 + 1 + 1 + 1

Candidate's Solution

Language used: Python 3

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left = left
        self.right = right
def root_equals_sum_of_descendants(root):
    # Write your code here
    def subtree_sum(node):
        if not node:
            return 0
        return node.val + subtree_sum(node.left) + subtree_sum(node.right)

    if not root:
        return True

    total = subtree_sum(root)
    return (total - root.val) == root.val

def list_to_tree(lst):
    if not lst:
        return None
    root = TreeNode(lst[0])
    queue = [root]
    i = 1
    while i < len(lst):
        current = queue.pop(0)
        if i < len(lst) and lst[i] is not None:
            current.left = TreeNode(lst[i])
            queue.append(current.left)
        i += 1
        if i < len(lst) and lst[i] is not None:
            current.right = TreeNode(lst[i])
```

```
42                 queue.append(current.right)
43             i += 1
44         return root
45 
46 if __name__ == '__main__':
47     outfile = open(os.environ['OUTPUT_PATH'], 'w')
48     input_data = sys.stdin.read().strip().splitlines()
49 
50     for data in input_data:
51 
52         root_list = ast.literal_eval(data)
53         root = list_to_tree(root_list)
54 
55         result = root_equals_sum_of_descendants(root)
56 
57         outfile.write(str(result) + '\n')
58     outfile.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Basic Case: True	Easy	Hidden	Success	0	0.0371 sec	10.9 KB
Basic Case: False	Easy	Hidden	Success	0	0.039 sec	10.9 KB
Tree with Missing Child	Easy	Hidden	Success	0	0.0283 sec	10.8 KB
Tree with Multiple Levels	Easy	Hidden	Success	0	0.0321 sec	10.9 KB
Single Node Tree	Easy	Hidden	Success	0	0.0563 sec	11 KB

Empty Tree	Easy	Hidden	Success	0	0.0332 sec	10.9 KB
Larger Tree	Easy	Hidden	Success	0	0.029 sec	10.9 KB
Tree with All Left Children	Easy	Hidden	Success	0	0.0311 sec	10.9 KB
Tree with Negative Values	Easy	Hidden	Success	0	0.0335 sec	10.9 KB
Tree with None in Non-Leaf Positions	Easy	Hidden	Success	0	0.0322 sec	10.9 KB
Balanced Tree	Easy	Hidden	Success	0	0.0323 sec	11 KB
Pass/Fail Case	Easy	Hidden	Success	20	0.0277 sec	10.9 KB

No comments.

4. Debug this code

Correct

Coding

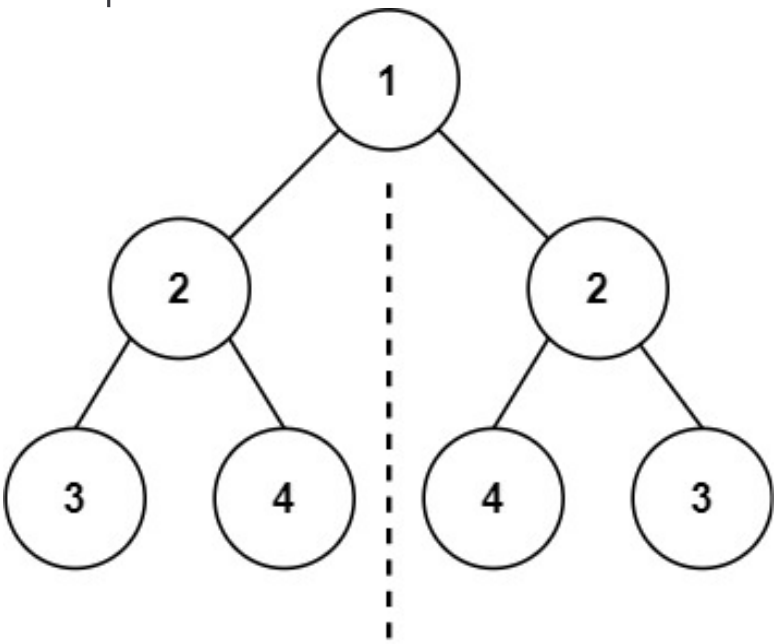
Question description

The following code incorrectly implements the function `is_symmetric()`. When implemented correctly, `is_symmetric()` accepts the `root` of a binary tree and

returns `True` if the tree is symmetric around its center and `False` otherwise.

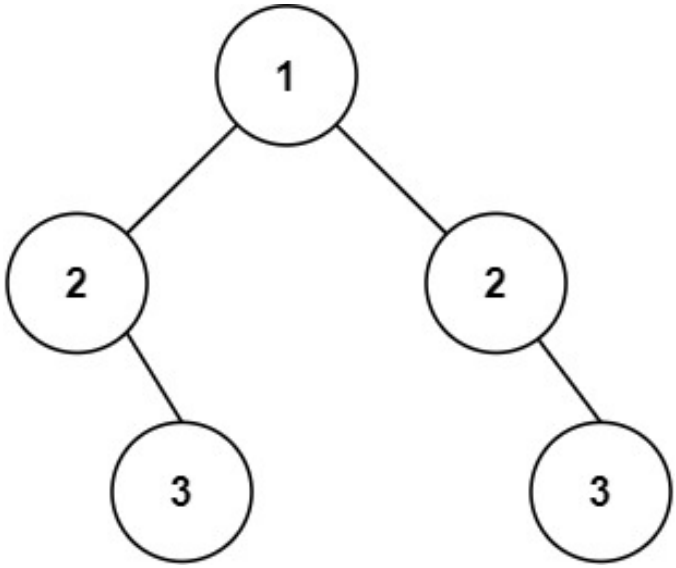
Identify any bug(s) within the given implementation and correct the code so that it successfully passes the provided test cases.

Example 1:



Input: root = [1,2,2,3,4,4,3]
Output: true

Example 2:



Input: root = [1,2,2,null,3,null,3]
Output: false

Candidate's Solution

Language used: **Python 3**

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import ast
from collections import deque

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def is_symmetric(root):
    def is_mirror(t1, t2):
        if not t1 and not t2:
```

```
            return True
        if not t1 or not t2:
            return False
        return (t1.val == t2.val and
                is_mirror(t1.left, t2.right) and
                is_mirror(t1.right, t2.left))

    return is_mirror(root.left, root.right)
def list_to_tree(lst):
    if not lst:
        return None

    root = TreeNode(lst[0])
    queue = deque([root])
    i = 1

    while i < len(lst):
        node = queue.popleft()

        # Handle the left child
        if lst[i] is not None:
            node.left = TreeNode(lst[i])
            queue.append(node.left)
        else:
            node.left = None
        i += 1

        # Handle the right child
        if i < len(lst):
            if lst[i] is not None:
                node.right = TreeNode(lst[i])
                queue.append(node.right)
            else:
                node.right = None
            i += 1

    return root

if __name__ == '__main__':
    input_data = sys.stdin.read().strip()
    input_list = ast.literal_eval(input_data)

    root = list_to_tree(input_list)

    result = is_symmetric(root)
    print(result)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Pass/Fail Case	Easy	Hidden	Success	5	0.0335 sec	11 KB

No comments.

5. What is the time complexity of mystery_function()?

Correct

Multiple Choice

Question description

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.value = val
        self.left_child = left
        self.right_child = right

def mystery_function(node):
    if not node:
        return 0

    left_result = mystery_function(node.left_child)
    right_result = mystery_function(node.right_child)

    return max(left_result, right_result) + 1
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- O(1)
- O(log n)
- **O(n)** ✓
- O(n log n)

No comments.

6. Which of the following best represents the values of the tree?

Correct

Multiple Choice

Question description

Given the following code, which of the following best represents the values of the tree with root `root`.

```
class TreeNode:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

a = Node('a')
x = Node('x')
y = Node('y')
e = Node('e')
m = Node('m')
```

```
p = Node('p')

a.left = x
a.right = y

x.left = e
x.right = m

y.right = p

root = a
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ <pre> <code> root / \ / \ x y / \ \ e m p</code></pre>
- ● **<pre> <code> a / \ / \ x y / \ \ e m p</code></pre>   ** ✓
- ○ <pre> <code> a / \ / \ x y / \ \ m e p</code></pre>
- ○ <pre> <code> a / \ / \ x y / \ / e m p </code></pre>

No comments.

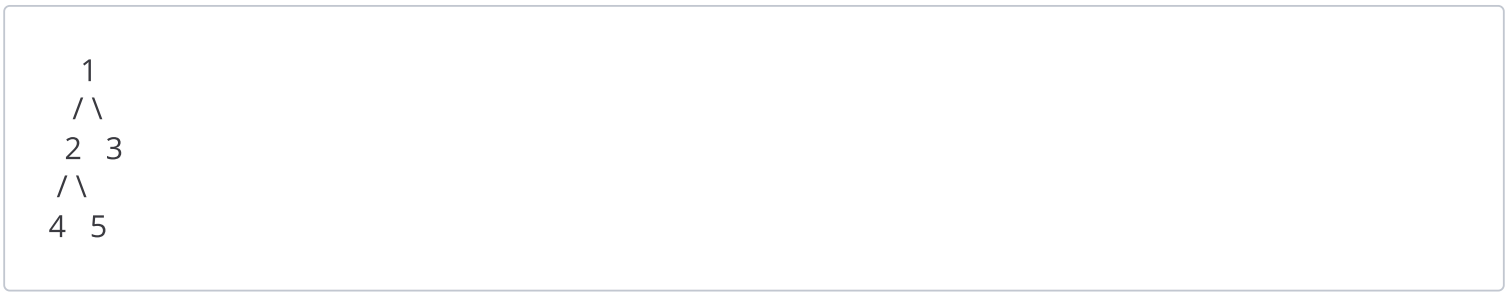
7. Which of the following arrays best represents its preorder traversal?

Correct

Multiple Choice

Question description

Given the following binary tree, which of the following arrays best represents its preorder traversal?



Candidate's Solution

Options: (Expected answer indicated with a tick)

- [ ] [4, 2, 5, 1, 3]
- [x] **[1, 2, 4, 5, 3]**
- [ ] [4, 5, 2, 3, 1]
- [ ] [1, 4, 2, 5, 3]

No comments.