

PDF generated at: 21 Jul 2024 21:33:59 UTC

View this report on HackerRank \Box

Score

100% • 80 / 80

scored in CodePath TIP101: Unit 6 Assessment, Version A - Summer 2024 in 74 min 7 sec on 21 Jul 2024 13:18:22 PDT

Candidate Information

Email concepting@protonmail.com

Test CodePath TIP101: Unit 6 Assessment, Version A - Summer 2024

Candidate Packet View ℃

Taken on 21 Jul 2024 13:18:22 PDT

Time taken 74 min 7 sec/ 90 min

Work Experience < 1 years

Invited by CodePath

Suspicious Activity detected

Code similarity



Code similarity

2 questions

Skill Distribution

Candidate Report Page 1 of 18



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

Status	No.	Question	Time Taken	Skill	Score
8	1	Tree Structure Multiple Choice	1 min 41 sec	-	5/5
8	2	Extract Even Values from a Singly Linked List Multiple Choice	1 min 38 sec	-	5/5

Candidate Report Page 2 of 18

⊗	3	Time Complexity Multiple Choice	11 min - 15 sec	5/5
⊗	4	Space Complexity Multiple Choice	33 sec	5/5
⊗	5	Find Nth From End (LL) Coding	11 min - 36 sec	20/20 🏳
⊗	6	Shuffle (LL) Coding	6 min 58 - sec	20/20
⊗	7	Find Intersection (LL) Coding	40 min 18 sec	20/20 🏳

1. Tree Structure

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')
```

Candidate Report Page 3 of 18

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

a.left = x
a.right = y

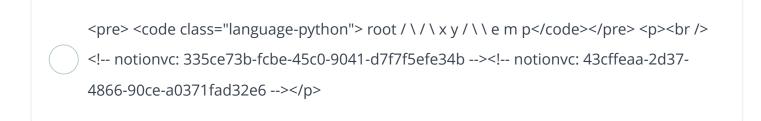
x.left = e
x.right = m

y.right = p

root = a
```

Candidate's Solution

Options: (Expected answer indicated with a tick)





<code class="language-python"> a / \ / \ x y / \ \ e m p</code>
 <!-- notionvc: 328084e6-4df2-4ae9-834b-5e008b6dce37 -->



<code class="language-python"> a / \ / \ x y / \ \ m e p</code>
 <!-notionvc: a3c29946-ba02-4e14-9e8b-f5aa1d00f0eb --><!-- notionvc: ad2c3bdc-8b34-4881996a-e480c9a25a6a --><!-- notionvc: 169fa24e-3d44-491e-a51c-1156fbd4cbc7 --><!-notionvc: 10453584-991a-4f8b-a411-722a914c99cb -->

Candidate Report Page 4 of 18

```
 <code class="language-python"> a / \ / \ x y / \ / e m p </code> <br /> <!--
notionvc: 9befdc5c-dc6d-4f6d-9268-e53ece276445 --><!-- notionvc: 04f9b35e-26c4-40e8-
95e0-4b1c535fd683 --><!-- notionvc: 28a7e321-eafd-4ce4-9c2e-1d3e932376dc --><!--
notionvc: 0832d073-fd74-457c-90db-735f13c730c7 --><!-- notionvc: a71bb310-8779-49dc-
9de1-6dbaafe45317 -->
```

No comments.

2. Extract Even Values from a Singly Linked List

⊘ Correct

Multiple Choice

Question description

What is the value of output?

Note: If output is a Node, the answer will include all nodes in the linked list represented by output.

```
class Node:
    def __init__(self, value, next_node=None):
        self.value = value
        self.next = next_node

def mystery_function(head):
    current = head
    output = []

while current:
    if current.value % 2 == 0:
        output.append(current.value)
    current = current.next

return output
```

Candidate Report Page 5 of 18

```
# Input List: 1 -> 2 -> 3 -> 4
head = Node(1, Node(2, Node(3, Node(4))))
output = mystery_function(head)
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

<code>[2, 4]</code>	\otimes
<code>2 -> 4</code>	
<code>[1, 3]</code>	
<code>1 -> 3</code>	
① No comments.	
ino confinients.	

3. Time Complexity

Multiple Choice

Question description

What is the time complexity of the following code? Assume n is the length of the linked list.

Candidate Report Page 6 of 18

```
def print_linked_list(head):
    current = head
    while current:
        print(current.value)
        current = current.next
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

O(1)	
O(n)	\otimes
O(n²)	
O(n³)	
No comments.	

4. Space Complexity

⊘ Correct

Multiple Choice

Question description

What is the space complexity of the following code? Assume n is the length of the linked list.

Candidate Report Page 7 of 18

```
def print_linked_list(head):
    current = head
    while current:
        print(current.value)
        current = current.next
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

O(1)	\otimes
O(n)	
O(n²)	
O(n³)	
① No comments.	

5. Find Nth From End (LL)

Correct

Coding

Question description

Candidate Report Page 8 of 18

Given a linked list, write a function **find_nth_from_end** that finds the nth node from the end of the list and returns its value.

```
# Example Input:
2
a->b->c->d->e->f

# Output:
e
```

Candidate's Solution

Language used: Python 3

```
1 #!/bin/python3
2
 3 import math
 4 import os
 5 import random
6 import re
7
   import sys
8
9
10
11 #
12 # Complete the 'find_nth_from_end' function below.
13
   #
14 # The function is expected to return a STRING.
15 # The function accepts following parameters:
      1. HEAD head
16 #
      2. INTEGER n
17 #
18 #
19
20 class Node:
21
      def init (self, value, next node = None):
22
          self.value = value
23
          self.next = next_node
24
25 def find_nth_from_end(head, n):
       # Write your code here
26
       # using two pointer approach
27
28
       point_one = head
29
       point_two = head
30
```

Candidate Report Page 9 of 18

```
31
        for i in range(n):
32
            if point one is None:
33
                return None
34
            point one = point_one.next
35
36
       while point one is not None:
37
            point one = point one.next
38
            point_two = point_two.next
39
40
       if point two is not None:
41
            return point two.value
42
        return None
43
44
   if name == ' main ':
45
       fptr = open(os.environ['OUTPUT_PATH'], 'w')
46
47
       # Helper function to convert str -> linked list
       def str_to_linked_list(vals_str):
48
            if vals str == "None":
49
50
                return None
            vals = vals str.split("->")
51
            temp head = Node("temp")
52
53
            temp curr = temp head
54
            for val in vals:
55
                temp curr.next = Node(val.strip())
                temp curr = temp curr.next
56
57
            return temp_head.next #Don't keep the temp head
58
59
       # Helper function to convert linked list to str
60
        def linked list to str(head):
            list_str = ""
61
62
            curr = head
63
            while curr:
                list str += curr.value
64
65
                if curr.next:
                    list str += "->"
66
67
                curr = curr.next
68
            if len(list str) == 0:
                return "None"
69
70
            return list str
71
72
       # Read and convert test input
73
       #inp = input()
       n = int(input())
74
75
       head = str to linked list(input())
76
```

Candidate Report Page 10 of 18

HackerRank ryan

```
# Call the function
77
       answer = find nth from end(head, n)
78
79
80
       # Turn the list into a string
       list_str = linked_list_to_str(head)
81
82
       # Bundle result in format <answer>|<linked list>
83
        result = str(answer) + "\n" + list_str
84
85
       fptr.write(result)
86
       fptr.close()
87
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	0	0.0338 sec	10.3 KB
Testcase 1	Easy	Hidden	Success	0	0.0318 sec	10.4 KB
Testcase 2	Easy	Hidden	Success	0	0.0373 sec	10.3 KB
Testcase 3	Easy	Hidden	Success	0	0.0329 sec	10.4 KB
Testcase 4	Easy	Hidden	Success	20	0.0383 sec	10.4 KB

! No comments.

6. Shuffle (LL)

⊘ Correct

Candidate Report Page 11 of 18

Language used: Python 3

Coding

Question description

Write a function that shuffles a linked list by swapping adjacent items. If there are an odd number of elements, the tail should not change position.

```
# Example 1:
# Input: a->b->c->d->e->f
# Output: b->a->d->c->f->e

# Notice that adjacent items have swapped position:
# a swapped with b
# c swapped with d
# e swapped with f
```

Candidate's Solution

17

2122

19 20

18 class Node:

23 def shuffle(head):

```
1 #!/bin/python3
 2
 3 import math
4 import os
 5 import random
 6 import re
7 import sys
8
9
10
11 #
12 # Complete the 'shuffle' function below.
13 #
14 # The function is expected to return nothing.
15 # The function accepts HEAD head as parameter.
16 #
```

Candidate Report Page 12 of 18

def init (self, value, next node = None):

self.value = value
self.next = next node

```
24
       # Write your code here
25
        current = head
26
27
       while current and current.next:
28
            current.value, current.next.value = current.next.value, current.value
29
            current = current.next.next
30
31
32
33
34
35
   if name == ' main ':
36
37
        fptr = open(os.environ['OUTPUT PATH'], 'w')
38
39
       # Helper function to convert str -> linked list
40
        def str to linked list(vals str):
            if vals str == "None":
41
42
                return None
43
            vals = vals str.split("->")
            temp head = Node("temp")
44
            temp curr = temp head
45
            for val in vals:
46
47
                temp curr.next = Node(val.strip())
48
                temp curr = temp curr.next
            return temp head.next #Don't keep the temp head
49
50
51
       # Helper function to convert linked list to str
52
        def linked list to str(head):
            list str = ""
53
            curr = head
54
55
            while curr:
                list str += curr.value
56
                if curr.next:
57
                    list str += "->"
58
59
                curr = curr.next
60
            if len(list str) == 0:
                return "None"
61
62
            return list str
63
64
       # Read and convert test input
65
       head = str to linked list(input())
66
67
       # Call the function
68
        answer = shuffle(head)
69
```

Candidate Report Page 13 of 18

```
70  # Turn the list back into a string
71  list_str = linked_list_to_str(head)
72
73  fptr.write(list_str)
74  fptr.close()
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	0	0.0364 sec	10.4 KB
Testcase 1	Easy	Hidden	Success	0	0.0393 sec	10.4 KB
Testcase 2	Easy	Hidden	Success	0	0.0353 sec	10.4 KB
Testcase 3	Easy	Hidden	Success	20	0.0398 sec	10.4 KB
Testcase 4	Easy	Hidden	Success	0	0.0382 sec	10.3 KB

No comments.

7. Find Intersection (LL)

Correct

ryan

Coding

Question description

Given the heads of two singly linked lists, return the node at which the two lists intersect. If the two linked lists do not intersect, return None. You may not modify either of the linked lists.

Candidate Report Page 14 of 18

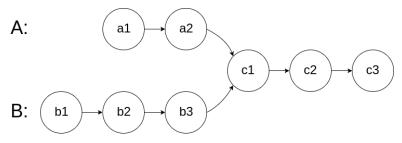
Language used: Python 3

```
EXAMPLE

list_a is a1->a2->c1->c2->c3

list_b is b1->b2->b3->c1->c2->c3

intersection point is c1
```



Example image from: https://leetcode.com/problems/intersection-of-two-linked-lists/description/

Candidate's Solution

```
1 #!/bin/python3
2
3 import math
4 import os
5 import random
6 import re
7 import sys
8
9
10
11 #
12 # Complete the 'find_intersection' function below.
```

```
13 #
14 # The function is expected to return a NODE.
15 # The function accepts following parameters:
     1. HEAD head a
16 #
17 #
      2. HEAD head b
18 #
19
20 class Node:
21
      def __init__(self, value, next_node = None):
22
          self.value = value
23
          self.next = next node
24
  def get_len(head):
25
```

Candidate Report Page 15 of 18

```
26
       length = 0
27
        current = head
28
       while current:
29
            lenath += 1
30
            current = current.next
31
        return length
32
33
   def find_intersection(head_a, head_b):
       # Write your code here
34
35
       length a = get len(head a)
36
       length_b = get_len(head_b)
37
38
       if length a > length b:
39
            for i in range(length a - length b):
40
                head a = head a.next
41
       else:
42
            for i in range(length b - length a):
43
                head b = head b.next
44
45
       while head a and head b:
46
            if head a == head b:
47
                return head a
48
            head a = head a.next
49
            head b = head b.next
        return None
50
51
52
53 if name == ' main ':
54
        fptr = open(os.environ['OUTPUT PATH'], 'w')
55
56
       # Helper function to convert str -> linked list
57
       def str to linked list without repetition(vals str, nodes dict={}):
58
            if vals str == "None":
                return None, {}
59
            vals = [x.strip() for x in vals_str.split("->")]
60
            temp head = Node("temp")
61
62
            temp curr = temp head
63
            for val in vals:
64
                if val in nodes dict:
65
                    temp curr.next = nodes dict[val]
66
                else:
67
                    temp curr.next = Node(val)
68
                    nodes dict[val] = temp curr.next
69
                temp curr = temp curr.next
70
            return temp head.next, nodes dict #Don't keep the temp head
71
```

Candidate Report Page 16 of 18

```
# Helper function to convert linked list to str
72
 73
         def linked list to str(head):
 74
             list str = ""
75
             curr = head
76
             while curr:
77
                 list str += curr.value
78
                 if curr.next:
                     list str += "->"
 79
                 curr = curr.next
 80
 81
             if len(list str) == 0:
 82
                 return "None"
 83
             return list str
 84
 85
        # Read and convert test input
        head_a, dict_a = str_to linked_list_without_repetition(input())
 86
        head_b, _ = str_to_linked_list_without_repetition(input(), dict a)
 87
 88
 89
        # Call the function
 90
        answer = find intersection(head a, head b)
 91
         if answer is not None:
 92
             answer = answer.value
 93
 94
        # Turn the lists into a string
 95
        list str a = linked list to str(head a)
        list str b = linked list to str(head b)
96
 97
98
        # Bundle result in format <answer>\n<original linked list>
99
         result = str(answer) + "\n" + list str a + "\n" + list str b
100
101
         fptr.write(result)
        fptr.close()
102
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden	Success	0	0.0478 sec	10.4 KB
Testcase 1	Easy	Hidden	Success	0	0.0365 sec	10.3 KB

Candidate Report Page 17 of 18

Testcase 2	Easy	Hidden	Success	0	0.0319 sec	10.3 KB
Testcase 3	Easy	Hidden	Success	20	0.032 sec	10.4 KB
Testcase 4	Easy	Hidden	Success	0	0.0356 sec	10.5 KB
Testcase 5	Easy	Hidden	Success	0	0.0355 sec	10.5 KB

No comments.

Candidate Report Page 18 of 18