



Full Name: Priyanka Jain
Email: psidjain123@gmail.com
Test Name: **Linked List Assessment**
Taken On: 10 Jun 2019 21:48:21 PDT
Time Taken: 89 min 56 sec/ 90 min
Personal Email Address: psidjain123@gmail.com
Invited by: Curriculum
Tags Score:

16.1%
155/965


scored in **Linked List Assessment** in 89 min 56 sec
on 10 Jun 2019 21:48:21 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Minimum Bytes Per Node > Multiple Choice	2 min 48 sec	5/ 5	✓
Q2	List Operations > Multiple Choice	1 min 5 sec	5/ 5	✓
Q3	Time and Space Complexity > Multiple Choice	7 min 58 sec	0/ 5	✗
Q4	Execution By Hand > Multiple Choice	7 min 33 sec	5/ 5	✓
Q5	Algorithm Space Complexity > Multiple Choice	1 min 52 sec	0/ 5	✗
Q6	Compute Length > Coding	2 min 41 sec	40/ 40	✓
Q7	Palindrome Linked List > Coding	38 min 31 sec	100/ 400	✓
Q8	Plus One Linked List > Coding	27 min 37 sec	0/ 400	✗
Q9	LRU Cache > Coding		0/ 100	⊖

QUESTION 1



Correct Answer

Score 5

Minimum Bytes Per Node > Multiple Choice

QUESTION DESCRIPTION


On a 64-bit machine, what is the minimum number of bytes per node needed to implement a Singly Linked List, assuming that each node stores a reference to its value?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

☐ 2


☐ 8

 ☒ 16

☐ 32

No Comments

QUESTION 2



Correct Answer

Score 5

List Operations > Multiple Choice

QUESTION DESCRIPTION


Given the list `1->2`, what would the result look like after the following operations are applied sequentially?

1. Insert(3)
2. Insert(4)
3. Delete(1)

What about after setting `head.next.next.val = 5`?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

 ☒ 4->3->2 4->3->5

☐ 2->3->4 3->3->5

☐ 2->4->3 2->5->3

☐ 2->4->1 2->5->1

No Comments

QUESTION 3



Wrong Answer

Score 0

Time and Space Complexity > Multiple Choice

QUESTION DESCRIPTION

What is the space and time complexity of the following algorithm for reversing a linked list?

```
def get_last(head):  
    if not head or not head.next:  
        return head  
    return get_last(head.next)  
  
def reverse(head):  
    if not head or not head.next:  
        return head  
    r = reverse(head.next)  
    l = get_last(r)  
    head.next = None  
    l.next = head  
    return r
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☒ Time Complexity: $O(n)$ Space Complexity: $O(1)$
- ☐ Time Complexity: $O(n)$ Space Complexity: $O(n)$
- ☐ Time Complexity: $O(n^2)$ Space Complexity: $O(1)$
- ☒ Time Complexity: $O(n^2)$ Space Complexity: $O(n^2)$

No Comments

QUESTION 4



Correct Answer

Score 5

Execution By Hand > Multiple Choice

QUESTION DESCRIPTION

What is the output of running the following code with the input `head = 1 → 2 → 3 → 4 → 5, k = 3`?

```
def do_what(head, k):  
    if not head:  
        return head  
  
    e = head  
    ne = head  
    i = 0  
    while i < k:  
        e = e.next  
        if not e:  
            return head  
        i += 1  
  
    while e.next:  
        ne = ne.next  
        e = e.next  
  
    d = Node("d")  
    d.next = ne.next  
    ne.next = None  
    e.next = head  
    return d.next
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☒ 3->4->5->1->2
- ☐ "e"->4->5->1->2
- ☐ "e"->1->2->3->4
- ☐ 5->1->2->3->4

No Comments

QUESTION 5



Wrong Answer

Score 0

Algorithm Space Complexity > Multiple Choice

QUESTION DESCRIPTION

What is the space complexity of the following algorithm for [splitting a linked list into parts](#)?

```
def splitListToParts(root, k):
    if k < 2:
        return [root]

    len_1 = 0
    c = root
    while c:
        len_1 += 1
        c = c.next
    binlen = int(len_1 / k)
    olen = len_1 - binlen * k
    blens = [binlen for i in range(k)]
    for i in range(olen):
        blens[i] += 1

    ds = [ListNode("dummy") for _ in range(k)]
    c = root
    t = 0
    b = 0
    cd = ds[0]
    while c:
        if t == blens[b]:
            b += 1
            t = 0
            cd = ds[b]
        cd.next = c
        c = c.next
        cd = cd.next
        cd.next = None
        t += 1
    return [d.next for d in ds]
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☒ ☐ O(k)
- ☒ ☐ O(n)
- ☐ ☐ O(n/k)
- ☐ ☐ O(n*k)

No Comments

QUESTION 6

✓

Correct Answer

Score 40

Compute Length > Coding

QUESTION DESCRIPTION

Please compute the length of the list A.

CANDIDATE ANSWER

Language used: Java 7

1

// Complete the getLength function below.

2

3

/*

4

* For your reference:

5

*

6

* SinglyLinkedListNode {

7

* int data;

8

* SinglyLinkedListNode next;

9

* }

10

*

11

*/

12

static int getLength(SinglyLinkedListNode A) {

13

SinglyLinkedListNode ptr = A;

14

int length = 0;

15

while(ptr != null){

16

length++;

17

ptr = ptr.next;

18

}

19

return length;

20

}

21

22

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.11 sec	23.6 MB
TestCase 1	Easy	Hidden case	✓ Success	10	0.113 sec	23.2 MB
TestCase 2	Easy	Hidden case	✓ Success	10	0.101 sec	23 MB
TestCase 3	Easy	Hidden case	✓ Success	10	0.112 sec	23.3 MB

No Comments

QUESTION 7

✓

Correct Answer

Score 100

Palindrome Linked List > Coding

QUESTION DESCRIPTION

Given a singly linked list, determine if it is a palindrome.

CANDIDATE ANSWER

Language used: Java 7

1

// Complete the isPalindrome function below.

2

3

/*

```

4      * For your reference:
5      *
6      * SinglyLinkedListNode {
7      *     int data;
8      *     SinglyLinkedListNode next;
9      * }
10     *
11     */
12 static boolean isPalindrome(SinglyLinkedListNode A) {
13     if( A == null || A.next == null){
14         return true;
15     }
16
17     // SinglyLinkedListNode ptr = A;
18     // int length = 0;
19     // while(ptr != null){
20     //     length++;
21     //     ptr = ptr.next;
22     // }
23
24     SinglyLinkedListNode slow = A;
25     SinglyLinkedListNode slow_prev = A;
26     SinglyLinkedListNode fast = A;
27     SinglyLinkedListNode middle = null; //for odd number of elements
28
29     //get the middle of the list
30     while(fast.next != null && fast != null){
31         fast = fast.next.next;
32         slow_prev = slow;
33         slow = slow.next;
34     }
35
36     //fast == null then eve number of elements
37     if(fast != null){
38         middle = slow;
39         slow = slow.next;
40         middle.next = null;
41     }
42
43     slow_prev.next = null;
44     SinglyLinkedListNode second_half = slow;
45     SinglyLinkedListNode first_half = A;
46     reverse(second_half);
47
48     while(first_half != null){
49         if(first_half.data != second_half.data)
50             return false;
51
52         first_half = first_half.next;
53         second_half = second_half.next;
54     }
55     return true;
56 }
57
58 static void reverse(SinglyLinkedListNode head){
59
60     SinglyLinkedListNode curr = head;
61     SinglyLinkedListNode prev = null;
62     SinglyLinkedListNode next = null;
63
64     while(curr != null){
65         //store next
66         next = curr.next;
67

```

```

68         //rev curr pointers node
69         curr.next = prev;
70
71         //increment pointer
72         prev = curr;
73         curr = next;
74     }
75     head = prev;
76     //return head;
77 }
78
79

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	10	0.104 sec	23.6 MB
TestCase 1	Easy	Hidden case	✔ Success	10	0.0938 sec	22.9 MB
TestCase 2	Easy	Hidden case	✘ Wrong Answer	0	0.096 sec	22.8 MB
TestCase 3	Easy	Hidden case	✔ Success	10	0.105 sec	23.4 MB
TestCase 4	Easy	Hidden case	✘ Wrong Answer	0	0.13 sec	24.1 MB
TestCase 5	Easy	Hidden case	✘ Wrong Answer	0	0.18 sec	25 MB
TestCase 6	Easy	Hidden case	✔ Success	10	0.108 sec	23.4 MB
TestCase 7	Easy	Hidden case	✘ Wrong Answer	0	0.128 sec	23.7 MB
TestCase 8	Easy	Hidden case	✘ Wrong Answer	0	0.123 sec	23.5 MB
TestCase 9	Easy	Hidden case	✘ Wrong Answer	0	0.119 sec	23.7 MB
TestCase 10	Easy	Hidden case	✘ Wrong Answer	0	0.196 sec	25.4 MB
TestCase 11	Easy	Hidden case	✘ Wrong Answer	0	0.198 sec	24.9 MB
TestCase 12	Easy	Hidden case	✘ Wrong Answer	0	0.183 sec	25 MB
TestCase 13	Easy	Hidden case	✘ Wrong Answer	0	0.192 sec	24.9 MB
TestCase 14	Easy	Hidden case	✔ Success	10	0.107 sec	23.7 MB
TestCase 15	Easy	Hidden case	✔ Success	10	0.127 sec	23.9 MB
TestCase 16	Easy	Hidden case	✘ Wrong Answer	0	0.219 sec	25.3 MB
TestCase 17	Easy	Hidden case	✘ Wrong Answer	0	0.211 sec	25 MB
TestCase 18	Easy	Hidden case	✘ Wrong Answer	0	0.212 sec	25 MB
TestCase 19	Easy	Hidden case	✔ Success	10	0.11 sec	23.3 MB
TestCase 20	Easy	Hidden case	✔ Success	10	0.117 sec	23.2 MB
TestCase 21	Easy	Hidden case	✘ Wrong Answer	0	0.134 sec	23.6 MB
TestCase 22	Easy	Hidden case	✘ Wrong Answer	0	0.119 sec	23.5 MB
TestCase 23	Easy	Hidden case	✘ Wrong Answer	0	0.207 sec	25.3 MB
TestCase 24	Easy	Hidden case	✘ Wrong Answer	0	0.169 sec	24.9 MB
TestCase 25	Easy	Hidden case	✘ Wrong Answer	0	0.22 sec	25.2 MB
TestCase 26	Easy	Hidden case	✘ Wrong Answer	0	0.146 sec	24.1 MB
TestCase 27	Easy	Hidden case	✘ Wrong Answer	0	0.167 sec	24.9 MB
TestCase 28	Easy	Hidden case	✘ Wrong Answer	0	0.14 sec	23.8 MB
TestCase 29	Easy	Hidden case	✘ Wrong Answer	0	0.129 sec	23.8 MB
TestCase 30	Easy	Hidden case	✘ Wrong Answer	0	0.148 sec	24.6 MB
TestCase 31	Easy	Hidden case	✘ Wrong Answer	0	0.13 sec	23.7 MB

TestCase 32	Easy	Hidden case	✗ Wrong Answer	0	0.19 sec	25 MB
TestCase 33	Easy	Hidden case	✗ Wrong Answer	0	0.221 sec	25.4 MB
TestCase 34	Easy	Hidden case	✗ Wrong Answer	0	0.227 sec	25.4 MB
TestCase 35	Easy	Hidden case	✗ Wrong Answer	0	0.198 sec	25.3 MB
TestCase 36	Easy	Hidden case	✓ Success	10	0.199 sec	25.2 MB
TestCase 37	Easy	Hidden case	✗ Wrong Answer	0	0.129 sec	23.6 MB
TestCase 38	Easy	Hidden case	✗ Wrong Answer	0	0.24 sec	25.7 MB
TestCase 39	Easy	Hidden case	✓ Success	10	0.198 sec	25.1 MB

No Comments

QUESTION 8



Wrong Answer

Score 0

Plus One Linked List > Coding

QUESTION DESCRIPTION

Given a non-negative integer represented as a non-empty singly linked list of digits, add one to the integer. You may assume the integer do not contain any leading zero, except the number 0 itself. The digits are stored such that the most significant digit is at the head of the list.

Example:

Input:
1->2->3

Output:
1->2->4

CANDIDATE ANSWER

The candidate did not manually submit any code. The last compiled version has been auto-submitted and the score you see below is for the auto-submitted version.

Language used: **Java 7**

```

1      // Complete the addOne function below.
2
3      /*
4       * For your reference:
5       *
6       * SinglyLinkedListNode {
7       *     int data;
8       *     SinglyLinkedListNode next;
9       * }
10     *
11     */
12     static SinglyLinkedListNode addOne(SinglyLinkedListNode A) {
13         SinglyLinkedListNode curr = A;
14         SinglyLinkedListNode temp = reverse(curr);
15         int carry = 0;
16
17         System.out.println(temp.data);

```

```

18     temp.data = (carry + temp.data + 1)%10;
19     carry = temp.data / 10;
20     temp = temp.next;
21
22     while(carry != 0 && temp != null){
23         System.out.println(temp.data);
24         temp.data = (carry + temp.data)%10;
25         System.out.println(temp.data);
26         carry = temp.data / 10;
27         temp = temp.next;
28     }
29
30     temp = reverse(temp);
31     return temp;
32 }
33
34 static SinglyLinkedListNode reverse(SinglyLinkedListNode head) {
35
36     SinglyLinkedListNode curr = head;
37     SinglyLinkedListNode prev = null;
38     SinglyLinkedListNode next = null;
39
40     while(curr != null){
41         //System.out.println("rev");
42         //store next
43         next = curr.next;
44
45         //rev curr pointers node
46         curr.next = prev;
47
48         //increment pointer
49         prev = curr;
50         curr = next;
51     }
52     head = prev;
53     return head;
54 }
55
56

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Wrong Answer	0	0.109 sec	23.3 MB
TestCase 1	Easy	Hidden case	Wrong Answer	0	0.101 sec	23.3 MB
TestCase 2	Easy	Hidden case	Wrong Answer	0	0.102 sec	23.3 MB
TestCase 3	Easy	Hidden case	Wrong Answer	0	0.111 sec	23.4 MB
TestCase 4	Easy	Hidden case	Wrong Answer	0	0.11 sec	23.8 MB
TestCase 5	Easy	Hidden case	Wrong Answer	0	0.148 sec	23.8 MB
TestCase 6	Easy	Hidden case	Wrong Answer	0	0.143 sec	23.9 MB
TestCase 7	Easy	Hidden case	Wrong Answer	0	0.101 sec	23 MB
TestCase 8	Easy	Hidden case	Wrong Answer	0	0.226 sec	25.5 MB
TestCase 9	Easy	Hidden case	Wrong Answer	0	0.295 sec	26.1 MB
TestCase 10	Easy	Hidden case	Wrong Answer	0	0.146 sec	24.1 MB
TestCase 11	Easy	Hidden case	Wrong Answer	0	0.197 sec	25.5 MB
TestCase 12	Easy	Hidden case	Wrong Answer	0	0.285 sec	26.1 MB
TestCase 13	Easy	Hidden case	Wrong Answer	0	0.121 sec	23.8 MB

TestCase 14	Easy	Hidden case	⊗ Wrong Answer	0	0.139 sec	23.9 MB
TestCase 15	Easy	Hidden case	⊗ Wrong Answer	0	0.109 sec	23.5 MB
TestCase 16	Easy	Hidden case	⊗ Wrong Answer	0	0.204 sec	25.1 MB
TestCase 17	Easy	Hidden case	⊗ Wrong Answer	0	0.213 sec	25.3 MB
TestCase 18	Easy	Hidden case	⊗ Wrong Answer	0	0.135 sec	23.7 MB
TestCase 19	Easy	Hidden case	⊗ Wrong Answer	0	0.133 sec	23.8 MB
TestCase 20	Easy	Hidden case	⊗ Wrong Answer	0	0.28 sec	26.1 MB
TestCase 21	Easy	Hidden case	⊗ Wrong Answer	0	0.261 sec	25.9 MB
TestCase 22	Easy	Hidden case	⊗ Wrong Answer	0	0.195 sec	25.3 MB
TestCase 23	Easy	Hidden case	⊗ Wrong Answer	0	0.193 sec	25.3 MB
TestCase 24	Easy	Hidden case	⊗ Wrong Answer	0	0.251 sec	25.7 MB
TestCase 25	Easy	Hidden case	⊗ Wrong Answer	0	0.118 sec	23.7 MB
TestCase 26	Easy	Hidden case	⊗ Wrong Answer	0	0.141 sec	23.7 MB
TestCase 27	Easy	Hidden case	⊗ Wrong Answer	0	0.272 sec	26.2 MB
TestCase 28	Easy	Hidden case	⊗ Wrong Answer	0	0.129 sec	23.7 MB
TestCase 29	Easy	Hidden case	⊗ Wrong Answer	0	0.284 sec	26.4 MB
TestCase 30	Easy	Hidden case	⊗ Wrong Answer	0	0.119 sec	23.7 MB
TestCase 31	Easy	Hidden case	⊗ Wrong Answer	0	0.265 sec	25.8 MB
TestCase 32	Easy	Hidden case	⊗ Wrong Answer	0	0.143 sec	23.9 MB
TestCase 33	Easy	Hidden case	⊗ Wrong Answer	0	0.144 sec	24 MB
TestCase 34	Easy	Hidden case	⊗ Wrong Answer	0	0.12 sec	23.9 MB
TestCase 35	Easy	Hidden case	⊗ Wrong Answer	0	0.141 sec	24 MB
TestCase 36	Easy	Hidden case	⊗ Wrong Answer	0	0.251 sec	26 MB
TestCase 37	Easy	Hidden case	⊗ Wrong Answer	0	0.137 sec	23.7 MB
TestCase 38	Easy	Hidden case	⊗ Wrong Answer	0	0.147 sec	23.7 MB
TestCase 39	Easy	Hidden case	⊗ Wrong Answer	0	0.148 sec	24.1 MB

No Comments

QUESTION 9



Not Submitted

Score 0

LRU Cache > Coding

QUESTION DESCRIPTION

Design and implement a data structure for [Least Recently Used \(LRU\) cache](#). It should support the following operations: `get` and `put`.

`get(key)` - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.

`put(key, value)` - Set or insert the value if the key is not already present. When the cache reached its capacity, it should invalidate the least recently used item before inserting a new item.

An optimal can do both operations in $O(1)$ time complexity.

Feel free to implement or use any data structures available in the standard library, unless you find a pre-built LRU Cache in the standard library.

Here is an example usage.

```
LRUCache cache = new LRUCache( 2 /* capacity */ );

cache.put(1, 1);
cache.put(2, 2);
cache.get(1);      // returns 1
cache.put(3, 3);    // evicts key 2
cache.get(2);       // returns -1 (not found)
cache.put(4, 4);    // evicts key 1
cache.get(1);       // returns -1 (not found)
cache.get(3);       // returns 3
cache.get(4);       // returns 4
```

CANDIDATE ANSWER



This candidate has not answered this question.

No Comments