You can view this report online at : https://www.hackerrank.com/x/tests/211896/candidates/26420404/report

Full Name:	My Nguyen
Email:	nguyen_my@yahoo.com
Test Name:	**Linked List Assessment**
Taken On:	6 Jun 2021 19:50:03 PDT
Time Taken:	35 min 23 sec/ 90 min
Personal Email Address:	nguyen_my@yahoo.com
Contact Number:	+14084096862
Resume:	https://hackerrank-resumes.s3.amazonaws.com/412894/JhbK9vK_4Bhc4Gvuv7s5hgcFJGeFCAThWIiNY1UGAfhwRPsrmVekT5ZtKXgX8QA2Ag/My_Nguyen_Resume.PDF
Linkedin:	https://www.linkedin.com/in/my-nguyen-87849
Invited by:	Curriculum
Skills Score:	
Tags Score:	

99%

955/965

scored in **Linked List Assessment** in 35 min 23 sec on 6 Jun 2021 19:50:03 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Minimum Bytes Per Node > **Multiple Choice**	52 sec	0/ 5	✗
Q2	List Operations > **Multiple Choice**	1 min 55 sec	5/ 5	✓
Q3	Time and Space Complexity > **Multiple Choice**	2 min 35 sec	0/ 5	✗
Q4	Execution By Hand > **Multiple Choice**	5 min 3 sec	5/ 5	✓
Q5	Algorithm Space Complexity > **Multiple Choice**	4 min 8 sec	5/ 5	✓
Q6	Compute Length > **Coding**	1 min 44 sec	40/ 40	✓
Q7	Palindrome Linked List > **Coding**	4 min 44 sec	400/ 400	✓
Q8	Plus One Linked List > **Coding**	6 min 56 sec	400/ 400	✓

Q9 LRU Cache › Coding 6 min 7 sec 100/ 100 ✓

QUESTION 1	Minimum Bytes Per Node › Multiple Choice
✗ Wrong Answer	
Score 0	

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	List Operations › Multiple Choice
✓ Correct Answer	
Score 5	

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

✓ Correct Answer

Score 5

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_l = 0
    c = root
    while c:
        len_l += 1
        c = c.next
    binlen = int(len_l / k)
    olen = len_l - 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**

```
    // Complete the getLength function below.

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */
    static int getLength(SinglyLinkedListNode A) {
        int count = 0;
        while (A != null) {
            count++;
            A = A.next;
        }
        return count;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.0603 sec	22 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.0575 sec	22.1 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.056 sec	22 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.0666 sec	22 KB

No Comments

QUESTION 7

✓ Correct Answer

Score 400

Palindrome Linked List › Coding

QUESTION DESCRIPTION

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

CANDIDATE ANSWER

Language used: **Java 7**

```
    // Complete the isPalindrome function below.

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
```

```
     *     SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */
    static boolean isPalindrome(SinglyLinkedListNode head) {
        SinglyLinkedListNode middle = findMiddle(head);
        SinglyLinkedListNode reversed = reverse(middle);
        while (head != null && reversed != null) {
            if (head.data != reversed.data)
                return false;
            head = head.next;
            reversed = reversed.next;
        }
        return true;
    }

    private static SinglyLinkedListNode findMiddle(SinglyLinkedListNode head)
{
        SinglyLinkedListNode slow = head;
        SinglyLinkedListNode fast = head;
        while (fast != null && fast.next != null) {
            fast = fast.next.next;
            slow = slow.next;
        }
        return slow;
    }

    private static SinglyLinkedListNode reverse(SinglyLinkedListNode head) {
        SinglyLinkedListNode previous = null;
        while (head != null) {
            SinglyLinkedListNode tmp = head.next;
            head.next = previous;
            previous = head;
            head = tmp;
        }
        return previous;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.0618 sec	22.2 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.0569 sec	22.1 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.0555 sec	21.9 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.0642 sec	22.1 KB
TestCase 4	Easy	Hidden case	✓ Success	10	0.0825 sec	23.7 KB
TestCase 5	Easy	Hidden case	✓ Success	10	0.0905 sec	25.6 KB
TestCase 6	Easy	Hidden case	✓ Success	10	0.0577 sec	22 KB
TestCase 7	Easy	Hidden case	✓ Success	10	0.0738 sec	24.1 KB
TestCase 8	Easy	Hidden case	✓ Success	10	0.0774 sec	23.5 KB
TestCase 9	Easy	Hidden case	✓ Success	10	0.0617 sec	22.8 KB
TestCase 10	Easy	Hidden case	✓ Success	10	0.096 sec	25.9 KB
TestCase 11	Easy	Hidden case	✓ Success	10	0.0939 sec	25.8 KB
TestCase 12	Easy	Hidden case	✓ Success	10	0.0869 sec	25.8 KB
TestCase 13	Easy	Hidden case	✓ Success	10	0.0979 sec	25.8 KB
TestCase 14	Easy	Hidden case	✓ Success	10	0.0594 sec	22 KB

TestCase 15	Easy	Hidden case	✓ Success	10	0.0732 sec	23.7 KB
TestCase 16	Easy	Hidden case	✓ Success	10	0.0901 sec	26.3 KB
TestCase 17	Easy	Hidden case	✓ Success	10	0.0807 sec	26 KB
TestCase 18	Easy	Hidden case	✓ Success	10	0.0993 sec	26.1 KB
TestCase 19	Easy	Hidden case	✓ Success	10	0.0618 sec	22 KB
TestCase 20	Easy	Hidden case	✓ Success	10	0.0656 sec	22.2 KB
TestCase 21	Easy	Hidden case	✓ Success	10	0.0794 sec	24.1 KB
TestCase 22	Easy	Hidden case	✓ Success	10	0.068 sec	22.8 KB
TestCase 23	Easy	Hidden case	✓ Success	10	0.0753 sec	25.9 KB
TestCase 24	Easy	Hidden case	✓ Success	10	0.0927 sec	25.6 KB
TestCase 25	Easy	Hidden case	✓ Success	10	0.0946 sec	26 KB
TestCase 26	Easy	Hidden case	✓ Success	10	0.0839 sec	24.8 KB
TestCase 27	Easy	Hidden case	✓ Success	10	0.0861 sec	25.8 KB
TestCase 28	Easy	Hidden case	✓ Success	10	0.0758 sec	24.9 KB
TestCase 29	Easy	Hidden case	✓ Success	10	0.0748 sec	24 KB
TestCase 30	Easy	Hidden case	✓ Success	10	0.0785 sec	24.8 KB
TestCase 31	Easy	Hidden case	✓ Success	10	0.0847 sec	24.1 KB
TestCase 32	Easy	Hidden case	✓ Success	10	0.0879 sec	25.5 KB
TestCase 33	Easy	Hidden case	✓ Success	10	0.1013 sec	26.2 KB
TestCase 34	Easy	Hidden case	✓ Success	10	0.0976 sec	26.2 KB
TestCase 35	Easy	Hidden case	✓ Success	10	0.0947 sec	25.9 KB
TestCase 36	Easy	Hidden case	✓ Success	10	0.0768 sec	25.8 KB
TestCase 37	Easy	Hidden case	✓ Success	10	0.072 sec	24 KB
TestCase 38	Easy	Hidden case	✓ Success	10	0.0968 sec	26.2 KB
TestCase 39	Easy	Hidden case	✓ Success	10	0.0973 sec	26 KB

No Comments

QUESTION 8

✓ Correct Answer

Score 400

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

Language used: **Java 7**

```java
    // Complete the addOne function below.

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */
    static SinglyLinkedListNode addOne(SinglyLinkedListNode head) {
        SinglyLinkedListNode reversed = reverse(head);
        SinglyLinkedListNode current = reversed;
        SinglyLinkedListNode previous = null;
        while (current != null) {
            if (current.data != 9) {
                current.data++;
                break;
            } else {
                current.data = 0;
                previous = current;
                current = current.next;
            }
        }
        if (current == null) {
            SinglyLinkedListNode node = new SinglyLinkedListNode(1);
            previous.next = node;
        }

        return reverse(reversed);
    }

    static SinglyLinkedListNode reverse(SinglyLinkedListNode head) {
        SinglyLinkedListNode previous = null;
        while (head != null) {
            SinglyLinkedListNode tmp = head.next;
            head.next = previous;
            previous = head;
            head = tmp;
        }
        return previous;
    }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.0586 sec	22.2 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.0543 sec	22.1 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.0632 sec	22 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.0617 sec	22.2 KB
TestCase 4	Easy	Hidden case	✓ Success	10	0.0578 sec	21.9 KB
TestCase 5	Easy	Hidden case	✓ Success	10	0.0747 sec	24 KB
TestCase 6	Easy	Hidden case	✓ Success	10	0.0779 sec	24 KB
TestCase 7	Easy	Hidden case	✓ Success	10	0.0534 sec	22 KB

TestCase 8	Easy	Hidden case	✓ Success	10	0.0842 sec	26.2 KB
TestCase 9	Easy	Hidden case	✓ Success	10	0.109 sec	26.6 KB
TestCase 10	Easy	Hidden case	✓ Success	10	0.0845 sec	24.3 KB
TestCase 11	Easy	Hidden case	✓ Success	10	0.0894 sec	26.2 KB
TestCase 12	Easy	Hidden case	✓ Success	10	0.1037 sec	26.4 KB
TestCase 13	Easy	Hidden case	✓ Success	10	0.0661 sec	22.4 KB
TestCase 14	Easy	Hidden case	✓ Success	10	0.0799 sec	23 KB
TestCase 15	Easy	Hidden case	✓ Success	10	0.0594 sec	22.1 KB
TestCase 16	Easy	Hidden case	✓ Success	10	0.0917 sec	25.8 KB
TestCase 17	Easy	Hidden case	✓ Success	10	0.0837 sec	26.5 KB
TestCase 18	Easy	Hidden case	✓ Success	10	0.0664 sec	23.3 KB
TestCase 19	Easy	Hidden case	✓ Success	10	0.0728 sec	23.3 KB
TestCase 20	Easy	Hidden case	✓ Success	10	0.1181 sec	27 KB
TestCase 21	Easy	Hidden case	✓ Success	10	0.1068 sec	26.5 KB
TestCase 22	Easy	Hidden case	✓ Success	10	0.0891 sec	25.9 KB
TestCase 23	Easy	Hidden case	✓ Success	10	0.0863 sec	26.1 KB
TestCase 24	Easy	Hidden case	✓ Success	10	0.0939 sec	26.3 KB
TestCase 25	Easy	Hidden case	✓ Success	10	0.0695 sec	22.3 KB
TestCase 26	Easy	Hidden case	✓ Success	10	0.0779 sec	24.3 KB
TestCase 27	Easy	Hidden case	✓ Success	10	0.1085 sec	26.9 KB
TestCase 28	Easy	Hidden case	✓ Success	10	0.0629 sec	22.4 KB
TestCase 29	Easy	Hidden case	✓ Success	10	0.1116 sec	26.7 KB
TestCase 30	Easy	Hidden case	✓ Success	10	0.0667 sec	22.3 KB
TestCase 31	Easy	Hidden case	✓ Success	10	0.1253 sec	26.6 KB
TestCase 32	Easy	Hidden case	✓ Success	10	0.0748 sec	24.1 KB
TestCase 33	Easy	Hidden case	✓ Success	10	0.0806 sec	23.9 KB
TestCase 34	Easy	Hidden case	✓ Success	10	0.0615 sec	22.3 KB
TestCase 35	Easy	Hidden case	✓ Success	10	0.0788 sec	24.1 KB
TestCase 36	Easy	Hidden case	✓ Success	10	0.111 sec	26.9 KB
TestCase 37	Easy	Hidden case	✓ Success	10	0.0789 sec	23.1 KB
TestCase 38	Easy	Hidden case	✓ Success	10	0.0715 sec	24 KB
TestCase 39	Easy	Hidden case	✓ Success	10	0.0729 sec	24.1 KB

No Comments

QUESTION 9

✓ Correct Answer

Score 100

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

Language used: **Java 7**

```java
static class LRUCache {
    private int capacity;
    private Map<Integer, Integer> map;

    public LRUCache(int capacity) {
        this.capacity = capacity;
        map = new LinkedHashMap<>();
    }

    public int get(int key) {
        if (map.containsKey(key)) {
            int value = map.get(key);
            map.remove(key);
            map.put(key, value);
            return value;
        } else {
            return -1;
        }
    }

    public void put(int key, int value) {
        if (map.containsKey(key))
            map.remove(key);
        else {
            if (map.size() >= capacity) {
                int k = map.keySet().iterator().next();
                map.remove(k);
            }
        }
        map.put(key, value);
    }
}
```

TESTCASE DIFFICULTY TYPE STATUS SCORE TIME TAKEN MEMORY USED

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.0789 sec	24.4 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.4251 sec	80.5 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.3278 sec	66.9 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.3877 sec	85.4 KB
TestCase 4	Easy	Hidden case	✓ Success	10	0.1899 sec	38.7 KB
TestCase 5	Easy	Hidden case	✓ Success	10	0.3782 sec	84.8 KB
TestCase 6	Easy	Hidden case	✓ Success	10	0.316 sec	58.9 KB
TestCase 7	Easy	Hidden case	✓ Success	10	0.3784 sec	74.9 KB
TestCase 8	Easy	Hidden case	✓ Success	10	0.3032 sec	58 KB
TestCase 9	Easy	Hidden case	✓ Success	10	0.2369 sec	46.2 KB

No Comments

PDF generated at: 7 Jun 2021 03:27:08 UTC