You can view this report online at : https://www.hackerrank.com/x/tests/1330874/candidates/40943465/report

Full Name:	Ruchit Bhardwaj
Email:	ruchitbh@usc.edu
Test Name:	**CodePath SE103: Unit 1 Assessment - Summer 2022**
Taken On:	7 Jun 2022 16:26:57 PDT
Time Taken:	83 min 2 sec/ 90 min
Personal Email Address:	ruchitbh@usc.edu
Invited by:	CodePath
Skills Score:	
Tags Score:	

88.6%
855/965

scored in **CodePath SE103: Unit 1 Assessment - Summer 2022** in 83 min 2 sec on 7 Jun 2022 16:26:57 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Minimum Bytes Per Node > **Multiple Choice**	3 min 24 sec	5/ 5	✓
Q2	List Operations > **Multiple Choice**	5 min 20 sec	0/ 5	⊗
Q3	Time and Space Complexity > **Multiple Choice**	3 min 34 sec	5/ 5	✓
Q4	Execution By Hand > **Multiple Choice**	5 min 31 sec	5/ 5	✓
Q5	Algorithm Space Complexity > **Multiple Choice**	3 min 10 sec	0/ 5	⊗
Q6	Compute Length > **Coding**	1 min 46 sec	40/ 40	✓
Q7	Palindrome Linked List > **Coding**	27 min 43 sec	400/ 400	✓
Q8	Plus One Linked List > **Coding**	25 min 32 sec	400/ 400	✓
Q9	LRU Cache > **Coding**	6 min 42 sec	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

✗ Wrong Answer

Score 0

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

✓ Correct Answer

Score 5

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_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 8**

```java
class Result {

    /*
     * Complete the 'getLength' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_SINGLY_LINKED_LIST A as parameter.
     */

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */

    public static int getLength(SinglyLinkedListNode A) {
        int count = 0;
        SinglyLinkedListNode curr = A;
        while (curr != null) {
            count++;
            curr = curr.next;
        }
        return count;
    }

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.1373 sec	29.3 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.1209 sec	29.8 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.1907 sec	29.4 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.1135 sec	29.5 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 8**

```java
class Result {

    /*
     * Complete the 'isPalindrome' function below.
     *
     * The function is expected to return a BOOLEAN.
     * The function accepts INTEGER_SINGLY_LINKED_LIST A as parameter.
     */

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */

    public static boolean isPalindrome(SinglyLinkedListNode head) {
        // printList(head);
        SinglyLinkedListNode reversedListHead = getReverseList(head);
        // printList(reversedListHead);
        SinglyLinkedListNode curr = head;
        SinglyLinkedListNode revCurr = reversedListHead;
        while (curr != null && revCurr != null) {
            if (curr.data != revCurr.data) {
                // System.out.println(curr.data + " and " + revCurr.data);
                return false;
            }
            curr = curr.next;
            revCurr = revCurr.next;
        }
        return true;
    }

    public static SinglyLinkedListNode getReverseList(SinglyLinkedListNode
head) {
        SinglyLinkedListNode prev = null;
        SinglyLinkedListNode curr = head;
        while (curr != null) {
            SinglyLinkedListNode nextNode = curr.next;
            curr.next = prev;
            prev = curr;
            curr = nextNode;
        }
        return prev;
    }

    // public static void printList(SinglyLinkedListNode head) {
    //     SinglyLinkedListNode curr = head;
    //     while (curr != null) {
    //         System.out.print(curr.data + "->\t");
    //         curr = curr.next;
    //     }
    //     System.out.println("\n");
    // }

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.1916 sec	29.7 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.1458 sec	29.4 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.1794 sec	29.6 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.1681 sec	29.5 KB
TestCase 4	Easy	Hidden case	✓ Success	10	0.1585 sec	29.8 KB
TestCase 5	Easy	Hidden case	✓ Success	10	0.2154 sec	30.1 KB
TestCase 6	Easy	Hidden case	✓ Success	10	0.1364 sec	29.9 KB
TestCase 7	Easy	Hidden case	✓ Success	10	0.1629 sec	29.7 KB
TestCase 8	Easy	Hidden case	✓ Success	10	0.1444 sec	29.6 KB
TestCase 9	Easy	Hidden case	✓ Success	10	0.208 sec	29.6 KB
TestCase 10	Easy	Hidden case	✓ Success	10	0.1848 sec	30 KB
TestCase 11	Easy	Hidden case	✓ Success	10	0.1386 sec	29.9 KB
TestCase 12	Easy	Hidden case	✓ Success	10	0.2031 sec	30 KB
TestCase 13	Easy	Hidden case	✓ Success	10	0.1614 sec	30.1 KB
TestCase 14	Easy	Hidden case	✓ Success	10	0.1274 sec	29.2 KB
TestCase 15	Easy	Hidden case	✓ Success	10	0.1403 sec	29.9 KB
TestCase 16	Easy	Hidden case	✓ Success	10	0.1518 sec	30.3 KB
TestCase 17	Easy	Hidden case	✓ Success	10	0.1546 sec	29.6 KB
TestCase 18	Easy	Hidden case	✓ Success	10	0.1538 sec	30.2 KB
TestCase 19	Easy	Hidden case	✓ Success	10	0.143 sec	29.5 KB
TestCase 20	Easy	Hidden case	✓ Success	10	0.1579 sec	29.7 KB
TestCase 21	Easy	Hidden case	✓ Success	10	0.1895 sec	29.6 KB
TestCase 22	Easy	Hidden case	✓ Success	10	0.1697 sec	29.7 KB
TestCase 23	Easy	Hidden case	✓ Success	10	0.2066 sec	30.1 KB
TestCase 24	Easy	Hidden case	✓ Success	10	0.1683 sec	30.1 KB
TestCase 25	Easy	Hidden case	✓ Success	10	0.1558 sec	30.4 KB
TestCase 26	Easy	Hidden case	✓ Success	10	0.2134 sec	29.7 KB
TestCase 27	Easy	Hidden case	✓ Success	10	0.1769 sec	29.8 KB
TestCase 28	Easy	Hidden case	✓ Success	10	0.158 sec	29.7 KB
TestCase 29	Easy	Hidden case	✓ Success	10	0.1695 sec	29.9 KB
TestCase 30	Easy	Hidden case	✓ Success	10	0.2325 sec	29.8 KB
TestCase 31	Easy	Hidden case	✓ Success	10	0.1979 sec	29.7 KB
TestCase 32	Easy	Hidden case	✓ Success	10	0.1677 sec	29.8 KB
TestCase 33	Easy	Hidden case	✓ Success	10	0.2316 sec	30.2 KB
TestCase 34	Easy	Hidden case	✓ Success	10	0.1561 sec	30.3 KB
TestCase 35	Easy	Hidden case	✓ Success	10	0.1547 sec	30.1 KB
TestCase 36	Easy	Hidden case	✓ Success	10	0.1427 sec	30.1 KB
TestCase 37	Easy	Hidden case	✓ Success	10	0.1321 sec	29.9 KB
TestCase 38	Easy	Hidden case	✓ Success	10	0.154 sec	30.3 KB

TestCase 38	Easy	Hidden case	✓ Success	10	0.154 sec	30.3 KB
TestCase 39	Easy	Hidden case	✓ Success	10	0.2013 sec	30 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 8**

```
class Result {

    /*
     * Complete the 'addOne' function below.
     *
     * The function is expected to return an INTEGER_SINGLY_LINKED_LIST.
     * The function accepts INTEGER_SINGLY_LINKED_LIST A as parameter.
     */

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */

    public static SinglyLinkedListNode addOne(SinglyLinkedListNode head) {
        SinglyLinkedListNode rev = reverseList(head);
        SinglyLinkedListNode curr = rev;
        int carry = 0;
        int sum = 0;
        int adder = 1;
        while (curr.next != null) {
            sum = curr.data + adder + carry;
            adder = 0;
            carry = (sum == 10) ? 1 : 0;
```

```java
            curr.data = sum % 10;
            curr = curr.next;
        }
        sum = curr.data + adder + carry;
        curr.data = sum % 10;
        if (sum == 10) {
            SinglyLinkedListNode lastNode = new SinglyLinkedListNode(1);
            curr.next = lastNode;
        }
        return reverseList(rev);
    }

    public static SinglyLinkedListNode reverseList(SinglyLinkedListNode head)
{
        SinglyLinkedListNode prev = null;
        SinglyLinkedListNode curr = head;
        while (curr != null) {
            SinglyLinkedListNode nextnode = curr.next;
            curr.next = prev;
            prev = curr;
            curr = nextnode;
        }
        return prev;
    }

    public static void printList(SinglyLinkedListNode head) {
        SinglyLinkedListNode curr = head;
        while (curr != null) {
            System.out.print(curr.data + "\t");
            curr = curr.next;
        }
        System.out.println();
    }
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	10	0.1171 sec	29.7 KB
TestCase 1	Easy	Hidden case	✓ Success	10	0.1312 sec	29.5 KB
TestCase 2	Easy	Hidden case	✓ Success	10	0.1308 sec	29.5 KB
TestCase 3	Easy	Hidden case	✓ Success	10	0.1622 sec	29.5 KB
TestCase 4	Easy	Hidden case	✓ Success	10	0.2041 sec	29.7 KB
TestCase 5	Easy	Hidden case	✓ Success	10	0.1553 sec	29.9 KB
TestCase 6	Easy	Hidden case	✓ Success	10	0.2134 sec	30 KB
TestCase 7	Easy	Hidden case	✓ Success	10	0.1503 sec	29.5 KB
TestCase 8	Easy	Hidden case	✓ Success	10	0.1615 sec	30.4 KB
TestCase 9	Easy	Hidden case	✓ Success	10	0.2167 sec	31.2 KB
TestCase 10	Easy	Hidden case	✓ Success	10	0.1799 sec	30 KB
TestCase 11	Easy	Hidden case	✓ Success	10	0.2013 sec	30.1 KB
TestCase 12	Easy	Hidden case	✓ Success	10	0.1602 sec	31.2 KB
TestCase 13	Easy	Hidden case	✓ Success	10	0.2138 sec	29.8 KB
TestCase 14	Easy	Hidden case	✓ Success	10	0.2038 sec	30 KB
TestCase 15	Easy	Hidden case	✓ Success	10	0.1735 sec	29.6 KB
TestCase 16	Easy	Hidden case	✓ Success	10	0.1615 sec	30.3 KB

TestCase 17	Easy	Hidden case	✓ Success	10	0.162 sec	30.3 KB
TestCase 18	Easy	Hidden case	✓ Success	10	0.1774 sec	30 KB
TestCase 19	Easy	Hidden case	✓ Success	10	0.1503 sec	30 KB
TestCase 20	Easy	Hidden case	✓ Success	10	0.1597 sec	30.8 KB
TestCase 21	Easy	Hidden case	✓ Success	10	0.1702 sec	31.6 KB
TestCase 22	Easy	Hidden case	✓ Success	10	0.2045 sec	30.4 KB
TestCase 23	Easy	Hidden case	✓ Success	10	0.15 sec	30.3 KB
TestCase 24	Easy	Hidden case	✓ Success	10	0.2375 sec	30.6 KB
TestCase 25	Easy	Hidden case	✓ Success	10	0.1413 sec	29 KB
TestCase 26	Easy	Hidden case	✓ Success	10	0.1668 sec	29.9 KB
TestCase 27	Easy	Hidden case	✓ Success	10	0.1651 sec	31.4 KB
TestCase 28	Easy	Hidden case	✓ Success	10	0.1863 sec	30.3 KB
TestCase 29	Easy	Hidden case	✓ Success	10	0.2072 sec	31.4 KB
TestCase 30	Easy	Hidden case	✓ Success	10	0.1302 sec	29.8 KB
TestCase 31	Easy	Hidden case	✓ Success	10	0.1576 sec	31.3 KB
TestCase 32	Easy	Hidden case	✓ Success	10	0.1424 sec	30.2 KB
TestCase 33	Easy	Hidden case	✓ Success	10	0.1825 sec	29.9 KB
TestCase 34	Easy	Hidden case	✓ Success	10	0.137 sec	29.6 KB
TestCase 35	Easy	Hidden case	✓ Success	10	0.1718 sec	30.1 KB
TestCase 36	Easy	Hidden case	✓ Success	10	0.2479 sec	31.5 KB
TestCase 37	Easy	Hidden case	✓ Success	10	0.1684 sec	29.9 KB
TestCase 38	Easy	Hidden case	✓ Success	10	0.1446 sec	30 KB
TestCase 39	Easy	Hidden case	✓ Success	10	0.1435 sec	30.3 KB

No Comments

QUESTION 9

Wrong Answer

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

Language used: **Java 8**

```java
static class LRUCache {
    public LRUCache(int capacity) {

    }

    public int get(int key) {

    }

    public void put(int key, int value) {

    }
}


```

Result: Compilation Failed

Compile Message

```
                  Solution.java:16: error: missing return statement
    }
    ^
1 error
```

No Comments

PDF generated at: 8 Jun 2022 00:51:23 UTC