

Name: Gobinda Narayan Pradhan

UID: 22BCS16759

Subject: Advance Programming Lab – II

Section: 22BCS-IoT_626 [B]

Subject Code: 22CSP-351

Date of Submission: 14th Feb

Assignment – 3

1. Print linked list

The screenshot shows a code editor with a dark theme. On the left, the 'Output Window' displays 'Compilation Results' for 'Y.O.G.I. (AI Bot)'. It states 'Problem Solved Successfully' with a green checkmark. Below this, it shows 'Test Cases Passed: 1112 / 1112', 'Attempts: Correct / Total: 2 / 2', 'Accuracy: 100%', and 'Time Taken: 1.83'. On the right, the code editor shows a Java class 'Solution' with a method 'printList(Node head)' that iterates through a linked list and prints each node's data.

2. Remove duplicates from a sorted list

The screenshot shows a code editor with a dark theme. On the left, the 'All Submissions' page for 'GobindNarayan' is displayed. It shows 'Accepted' status with '168 / 168 testcases passed'. The submission was made on Feb 14, 2025 at 10:26. Performance metrics are shown: Runtime 0 ms (Beats 100.00%), Memory 44.16 MB (Beats 55.25%). A bar chart shows the runtime performance across different test cases. On the right, the code editor shows a Java class 'Solution' with a method 'deleteDuplicates(ListNode head)' that removes duplicates from a sorted linked list. Below the code, the 'Test Result' section shows 'Accepted' status with 'Runtime: 0 ms'. The input is 'head = [1,1,2]' and the output is empty.

3. [Reverse a Linked List](#)

← All Submissions

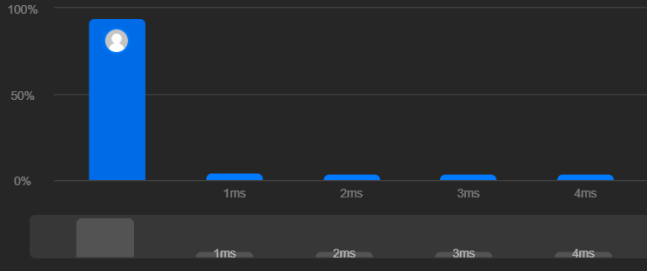
Accepted 28 / 28 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:31

Editorial Solution

Runtime: 0 ms | Beats 100.00% | Memory: 42.36 MB | Beats 81.18%

Analyze Complexity



Code | Java

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode dummy = new ListNode();
        while (head != null) {
            ListNode next = head.next;
            head.next = dummy.next;
            dummy.next = head;
            head = next;
        }
        return dummy.next;
    }
}
```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head = [1,2,3,4,5]

4. [Delete middle node of a list](#)

← All Submissions

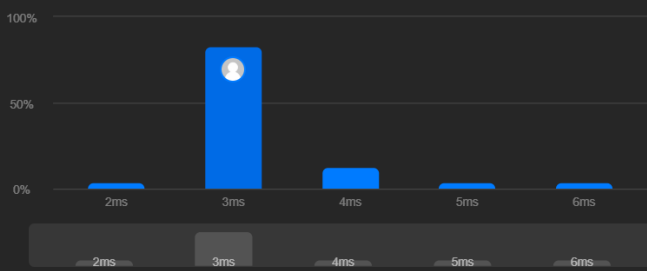
Accepted 70 / 70 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:33

Editorial Solution

Runtime: 3 ms | Beats 99.75% | Memory: 65.91 MB | Beats 5.60%

Analyze Complexity



Code | Java

```
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        ListNode dummy = new ListNode(0, head);
        ListNode slow = dummy, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        slow.next = slow.next.next;
        return dummy.next;
    }
}
```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head = [1,3,4,7,1,2,6]

5. Merge two sorted linked lists

All Submissions

Accepted 208 / 208 testcases passed
GobindNarayan submitted at Feb 14, 2025 10:34

Editorial Solution

Runtime: 0 ms | Beats 100.00%
Memory: 42.48 MB | Beats 73.62%

Analyze Complexity

Code | Java

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        if (list1 == null) {
            return list2;
        }
        if (list2 == null) {
            return list1;
        }
        if (list1.val <= list2.val) {
            list1.next = mergeTwoLists(list1.next, list2);
            return list1;
        } else {
            list2.next = mergeTwoLists(list1, list2.next);
            return list2;
        }
    }
}
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

list1 =
[1,2,4]

6. Remove duplicates from sorted lists 2

All Submissions

Accepted 166 / 166 testcases passed
GobindNarayan submitted at Feb 14, 2025 10:35

Editorial Solution

Runtime: 0 ms | Beats 100.00%
Memory: 43.03 MB | Beats 83.48%

Analyze Complexity

Code | Java

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        ListNode dummy = new ListNode(0, head);
        ListNode pre = dummy;
        ListNode cur = head;
        while (cur != null) {
            while (cur.next != null && cur.next.val == cur.val) {
                cur = cur.next;
            }
            if (pre.next == cur) {
                pre = cur;
            } else {
                pre.next = cur.next;
            }
            cur = cur.next;
        }
    }
}
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =
[1,2,3,3,4,4,5]

7. [Detect a cycle in a linked list](#)

← All Submissions

Accepted 29 / 29 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:36

Editorial Solution

Runtime 4 ms | Beats 13.36%

Memory 45.48 MB | Beats 7.26%

Analyze Complexity

Code | Java

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        Set<ListNode> s = new HashSet<>();
        for (; head != null; head = head.next) {
            if (!s.add(head)) {
                return true;
            }
        }
        return false;
    }
}
```

Java Auto

```
1 public class Solution {
2     public boolean hasCycle(ListNode head) {
3         Set<ListNode> s = new HashSet<>();
4         for (; head != null; head = head.next) {
5             if (!s.add(head)) {
6                 return true;
7             }
8         }
9         return false;
10    }
11 }
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =
[3,2,0,-4]

8. [Reverse linked list 2](#)

← All Submissions

Accepted 44 / 44 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:38

Editorial Solution

Runtime 0 ms | Beats 100.00%

Memory 41.63 MB | Beats 19.45%

Analyze Complexity

Code | Java

```
class Solution {
    public ListNode reverseBetween(ListNode head, int left, int right) {
        if (head.next == null || left == right) {
            return head;
        }
        ListNode dummy = new ListNode(0, head);
        ListNode pre = dummy;
        for (int i = 0; i < left - 1; ++i) {
            pre = pre.next;
        }
        ListNode p = pre;
        ListNode q = pre.next;
        ListNode cur = q;
        for (int i = 0; i < right - left + 1; ++i) {
            cur = cur.next;
        }
        q.next = null;
        cur.next = p;
        return dummy.next;
    }
}
```

Java Auto

```
1 class Solution {
2     public ListNode reverseBetween(ListNode head, int left, int right) {
3         if (head.next == null || left == right) {
4             return head;
5         }
6         ListNode dummy = new ListNode(0, head);
7         ListNode pre = dummy;
8         for (int i = 0; i < left - 1; ++i) {
9             pre = pre.next;
10        }
11        ListNode p = pre;
12        ListNode q = pre.next;
13        ListNode cur = q;
14        for (int i = 0; i < right - left + 1; ++i) {
15            cur = cur.next;
16        }
17        q.next = null;
18        cur.next = p;
19        return dummy.next;
20    }
21 }
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =
[1,2,3,4,5]

9. [Rotate a list](#)

← All Submissions

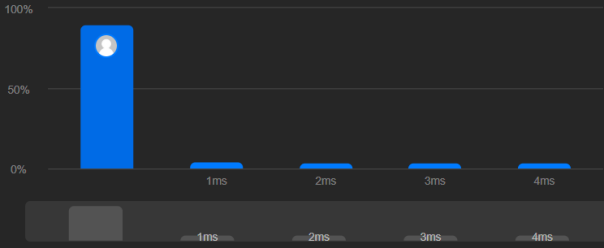
Accepted 232 / 232 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:39

Editorial Solution

Runtime 0 ms | Beats 100.00%
Memory 42.89 MB | Beats 25.71%

Analyze Complexity



Code | Java

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null) {
            return head;
        }
        ListNode cur = head;
        int n = 0;
        for (; cur != null; cur = cur.next) {
            n++;
        }
        k %= n;
        if (k == 0) {
            return head;
        }
        ListNode fast = head;
        ListNode slow = head;
```

Java Auto

```
1 class Solution {
2     public ListNode rotateRight(ListNode head, int k) {
3         if (head == null || head.next == null) {
4             return head;
5         }
6         ListNode cur = head;
7         int n = 0;
8         for (; cur != null; cur = cur.next) {
9             n++;
10        }
11        k %= n;
12        if (k == 0) {
13            return head;
14        }
15        ListNode fast = head;
16        ListNode slow = head;
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =
[1,2,3,4,5]

10. [Sort List](#)

← All Submissions

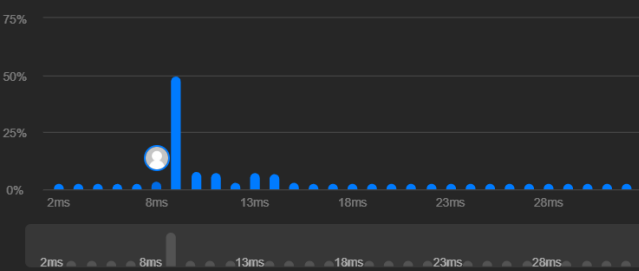
Accepted 30 / 30 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:40

Editorial Solution

Runtime 8 ms | Beats 99.30%
Memory 57.29 MB | Beats 22.70%

Analyze Complexity



Code | Java

```
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }
        ListNode slow = head, fast = head.next;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        ListNode l1 = head, l2 = slow.next;
        slow.next = null;
        l1 = sortList(l1);
        l2 = sortList(l2);
```

Java Auto

```
1 class Solution {
2     public ListNode sortList(ListNode head) {
3         if (head == null || head.next == null) {
4             return head;
5         }
6         ListNode slow = head, fast = head.next;
7         while (fast != null && fast.next != null) {
8             slow = slow.next;
9             fast = fast.next.next;
10        }
11        ListNode l1 = head, l2 = slow.next;
12        slow.next = null;
13        l1 = sortList(l1);
14        l2 = sortList(l2);
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =
[4,2,1,3]



11. Detect a cycle in a linked list 2

← All Submissions

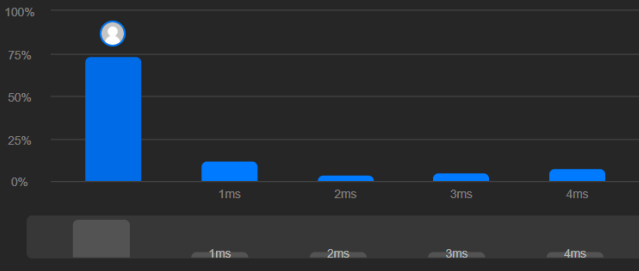
Accepted 18 / 18 testcases passed

GobindNarayan submitted at Feb 14, 2025 10:41

Editorial Solution

Runtime 0 ms | Beats 100.00%  **Memory** 44.45 MB | Beats 82.31% 

Analyze Complexity



Code | Java

```
1 public class Solution {
2     public ListNode detectCycle(ListNode head) {
3         ListNode fast = head, slow = head;
4         while (fast != null && fast.next != null) {
5             slow = slow.next;
6             fast = fast.next.next;
7             if (slow == fast) {
8                 ListNode ans = head;
9                 while (ans != slow) {
10                     ans = ans.next;
11                     slow = slow.next;
12                 }
13                 return ans;
14             }
15         }
16     }
17 }
```

Saved

☒ Testcase ☐ Test Result

Case 1 Case 2 Case 3 +

head =

[3,2,0,-4]