

AP- ASSIGNMENT

1. <https://www.geeksforgeeks.org/problems/print-linked-list-elements/0>

The screenshot shows the GeeksforGeeks interface for the problem 'Print Linked List Elements'. The 'Output Window' is open, displaying 'Compilation Results' and 'Problem Solved Successfully'. The 'Test Cases Passed' section shows '1112 / 1112'. The 'Attempts : Correct / Total' section shows '2 / 2'. The 'Accuracy : 100%' is also displayed. The 'Time Taken' section shows '0.09'. The code editor on the right shows a C++ solution for printing a linked list.

```
1 // } Driver Code Ends
19
20
21 class Solution {
22 public:
23 // Function to display the elements of a linked list in same line
24 void printList(Node *head) {
25 // your code goes here
26 Node* temp=head; // temp ptr pointing to head
27 while(temp!=NULL){ //traverse till the last element
28 cout<<temp->data<<" "; //print each element
29 temp=temp->next; // increment temp ptr
30 }
31 }
32 };
33
34 // } Driver Code Ends
```

2. <https://leetcode.com/problems/remove-duplicates-from-sorted-list/submissions/>

The screenshot shows the LeetCode interface for the problem 'Remove Duplicates from Sorted List'. The 'Accepted' status is shown with '168 / 168 testcases passed'. The 'Runtime' section shows '0 ms' and 'Beats 100.00%'. The 'Memory' section shows '16.02 MB' and 'Beats 90.25%'. A bar chart shows the performance distribution. The code editor on the right shows a C++ solution for removing duplicates from a sorted linked list.

```
7 ListNode* prev = head;
8 ListNode* temp = head->next;
9
10 while(temp!=NULL){
11 if(prev->val==temp->val){
12 temp=temp->next;
13 }else{
14 prev->next=temp;
15 prev=temp;
16 temp=temp->next;
17 }
18 }
19
20 prev->next=temp;
21 prev=temp;
22
23 return head;
24
25
26
27 };
```

3. <https://leetcode.com/problems/reverse-linked-list/description/>

The screenshot shows the LeetCode interface for the 'Reverse Linked List' problem. The submission is 'Accepted' with 28/28 testcases passed. The runtime is 0 ms, beating 100.00% of other submissions. The memory usage is 13.46 MB, beating 40.49%. A bar chart shows the runtime distribution, with the majority of submissions falling between 0ms and 1ms. The code is written in C++ and implements a recursive solution to reverse the linked list.

Runtime: 0 ms | Beats 100.00%
Memory: 13.46 MB | Beats 40.49%

```
1 class Solution {
2 public:
3     ListNode* reverseList(ListNode* head) {
4         ListNode* prev = NULL;
5         ListNode* curr = head;
6
7         while (curr != NULL) {
8             ListNode* nextTemp = curr->next;
9             curr->next = prev;
10            prev = curr;
11            curr = nextTemp;
12        }
13        return prev;
14    }
15};
```

4. <https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/submissions/>

The screenshot shows the LeetCode interface for the 'Delete the Middle Node of a Linked List' problem. The submission is 'Accepted' with 70/70 testcases passed. The runtime is 3 ms, beating 45.75% of other submissions. The memory usage is 311.88 MB, beating 98.45% of other submissions. A bar chart shows the runtime distribution, with most submissions falling between 0ms and 2ms. The code is written in C++ and implements a solution to delete the middle node of a singly-linked list.

Runtime: 3 ms | Beats 45.75%
Memory: 311.88 MB | Beats 98.45%

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if(head->next==NULL) return nullptr;
        ListNode* slow=head;
        ListNode* fast=head;
        ListNode* prev=head;
        while(fast->next->next!=NULL){
            prev=slow;
            slow=slow->next;
            fast=fast->next->next;
        }
        prev->next=slow->next;
        delete slow;
        return head;
    }
};
```


5. <https://leetcode.com/problems/merge-two-sorted-lists/submissions/1542046736/>

Accepted 208 / 208 testcases passed
smsharma submitted at Feb 14, 2025 17:12

Editorial Solution

Runtime
0 ms | Beats 100.00% 🏆
Analyze Complexity

Memory
19.43 MB | Beats 62.56% 🏆



```
1 class Solution {
2 public:
3     ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
4         if (!list1 || !list2)
5             return list1 ? list1 : list2;
6         if (list1->val > list2->val)
7             swap(list1, list2);
8         list1->next = mergeTwoLists(list1->next, list2);
9         return list1;
10    }
11};
```

Saved

Testcase Test Result

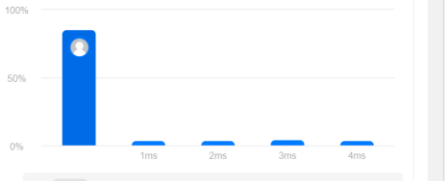
6. <https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/>

Accepted 166 / 166 testcases passed
smsharma submitted at Feb 14, 2025 17:13

Editorial Solution

Runtime
0 ms | Beats 100.00% 🏆
Analyze Complexity

Memory
15.54 MB | Beats 92.71% 🏆



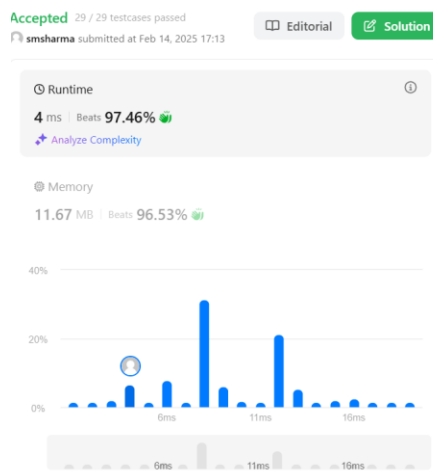
```
1 class Solution {
2 public:
3     ListNode* deleteDuplicates(ListNode* head) {
4         ListNode dummy(0, head);
5         ListNode* prev = &dummy;
6
7         while (head != nullptr) {
8             while (head->next && head->val == head->next->val)
9                 head = head->next;
10            if (prev->next == head)
11                prev = prev->next;
12            else
13                prev->next = head->next;
14            head = head->next;
15        }
16        return dummy.next;
17    }
18};
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

7. <https://leetcode.com/problems/linked-list-cycle/description/>



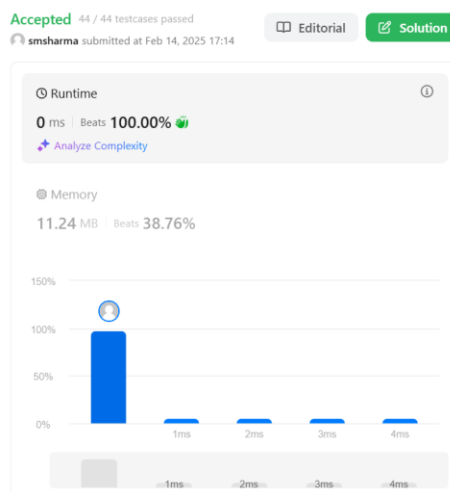
```
1 //  
2 class Solution {  
3 public:  
4     bool hasCycle(ListNode* head) {  
5         ListNode* fast = head;  
6         ListNode* slow = head;  
7  
8         while(fast != NULL && fast->next != NULL){  
9             fast = fast->next->next;  
10            slow = slow->next;  
11  
12            if(fast == slow){  
13                return true;  
14            }  
15        }  
16        return false;  
17    }  
18 }  
19
```

Saved

Testcase > Test Result

Accepted Runtime: 0 ms

8. <https://leetcode.com/problems/reverse-linked-list-ii/>



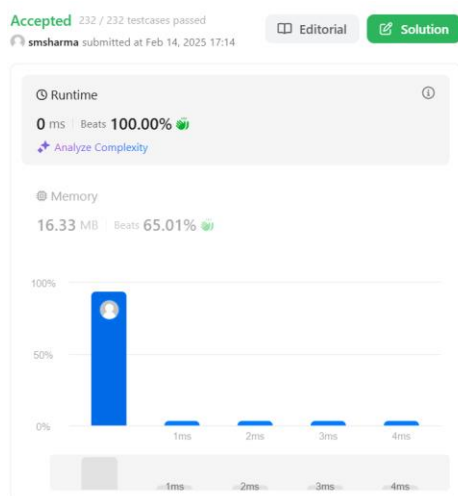
```
1 class Solution {  
2 public:  
3     ListNode* reverseBetween(ListNode* head, int left, int right) {  
4         if (left == 1)  
5             return reverseList(head, right);  
6         head->next = reverseBetween(head->next, left - 1, right - 1);  
7         return head;  
8     }  
9  
10 private:  
11     ListNode* reverseList(ListNode* head, int n) {  
12         if (n == 1)  
13             return head;  
14  
15         ListNode* newHead = reverseList(head->next, n - 1);  
16         ListNode* headNext = head->next;  
17         head->next = headNext->next;  
18         headNext->next = head;  
19         return newHead;  
20     }  
21 }  
22
```

Saved

Testcase > Test Result

Accepted Runtime: 0 ms

9. <https://leetcode.com/problems/rotate-list/>



```
1 class Solution {  
2 public:  
3     ListNode* rotateRight(ListNode* head, int k) {  
4         if (!head || !head->next || k == 0)  
5             return head;  
6  
7         ListNode* tail;  
8         int length = 1;  
9         for (tail = head; tail->next; tail = tail->next)  
10            ++length;  
11         tail->next = head; // Circle the list.  
12  
13         const int t = length - k % length;  
14         for (int i = 0; i < t; ++i)  
15             tail = tail->next;  
16         ListNode* newHead = tail->next;  
17         tail->next = nullptr;  
18  
19         return newHead;  
20     }  
21 }  
22
```

Saved

Testcase > Test Result

Accepted Runtime: 0 ms

10. <https://leetcode.com/problems/sort-list/submissions/1542851463/>

Accepted 30 / 30 testcases passed
smsharma submitted at Feb 14, 2025 17:15

Editorial

Solution

Runtime

12 ms | Beats 78.70%

Analyze Complexity

Memory

57.05 MB | Beats 86.19%

Analyze Complexity



```

30 | head->next = nullptr;
31 | return rest;
32 | }
33 |
34 | pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {
35 |     ListNode dummy(0);
36 |     ListNode* tail = &dummy;
37 |
38 |     while (l1 && l2) {
39 |         if (l1->val > l2->val)
40 |             swap(l1, l2);
41 |         tail->next = l1;
42 |         l1 = l1->next;
43 |         tail = tail->next;
44 |     }
45 |     tail->next = l1 ? l1 : l2;
46 |     while (tail->next != nullptr)
47 |         tail = tail->next;
48 |     return {dummy.next, tail};
49 | }
50 |
51 |
52 |
53 |
54 |
55 |
56 |
57 |

```

Saved

Ln 57,

Testcase Test Result

Accepted Runtime: 0 ms

11. <https://leetcode.com/problems/linked-list-cycle-ii/description/>

Accepted 30 / 30 testcases passed
smsharma submitted at Feb 14, 2025 17:15

Editorial

Solution

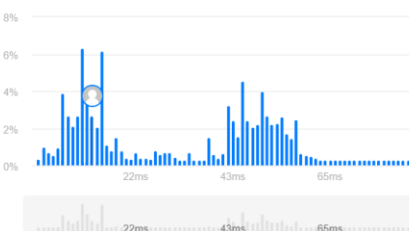
Runtime

12 ms | Beats 78.70%

Analyze Complexity

Memory

57.05 MB | Beats 86.19%



```

36 | head->next = nullptr;
37 | return rest;
38 | }
39 |
40 | pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {
41 |     ListNode dummy(0);
42 |     ListNode* tail = &dummy;
43 |
44 |     while (l1 && l2) {
45 |         if (l1->val > l2->val)
46 |             swap(l1, l2);
47 |         tail->next = l1;
48 |         l1 = l1->next;
49 |         tail = tail->next;
50 |     }
51 |     tail->next = l1 ? l1 : l2;
52 |     while (tail->next != nullptr)
53 |         tail = tail->next;
54 |     return {dummy.next, tail};
55 | }
56 |
57 |

```

Saved

Testcase Test Result

Accepted Runtime: 0 ms