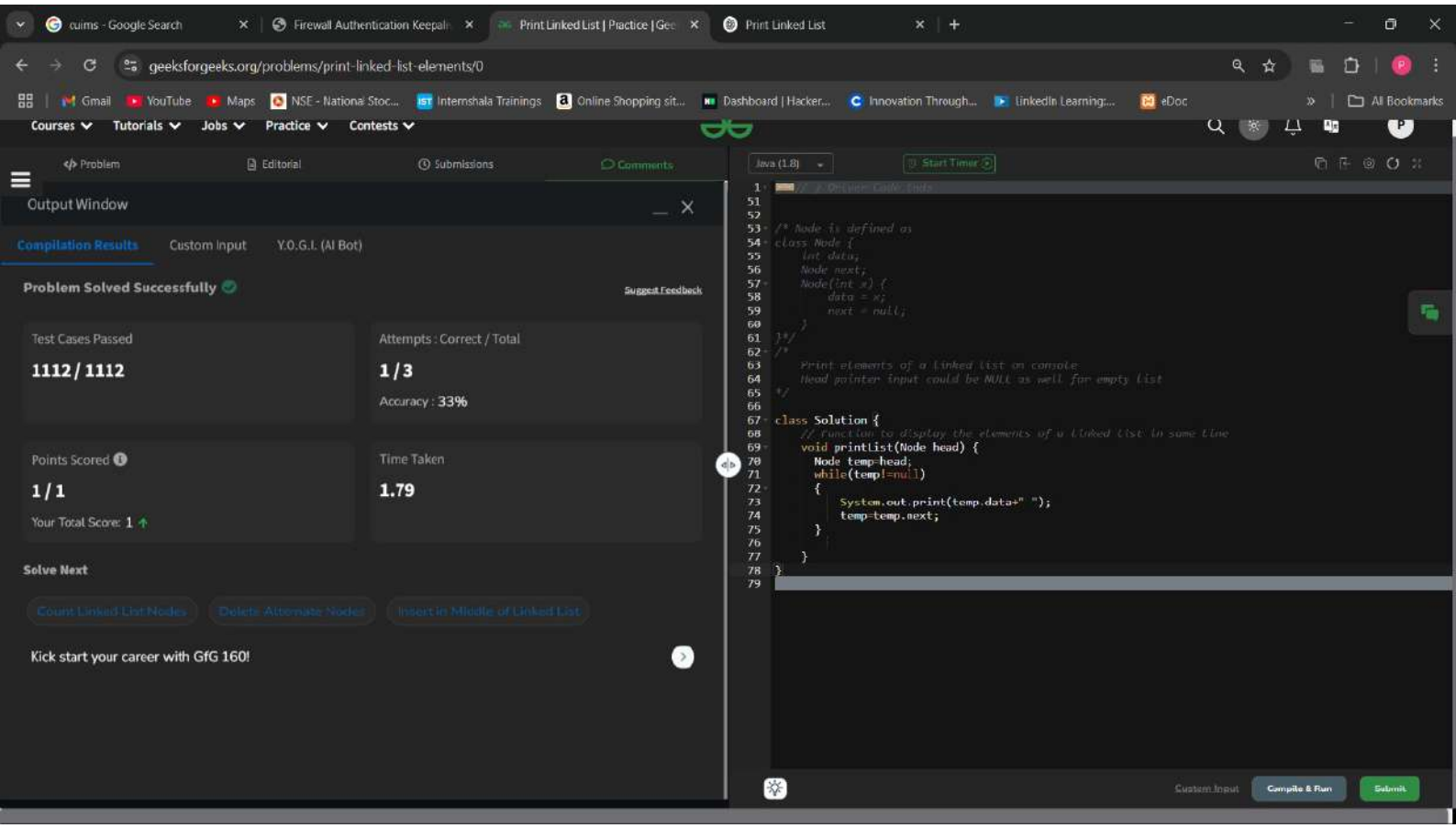
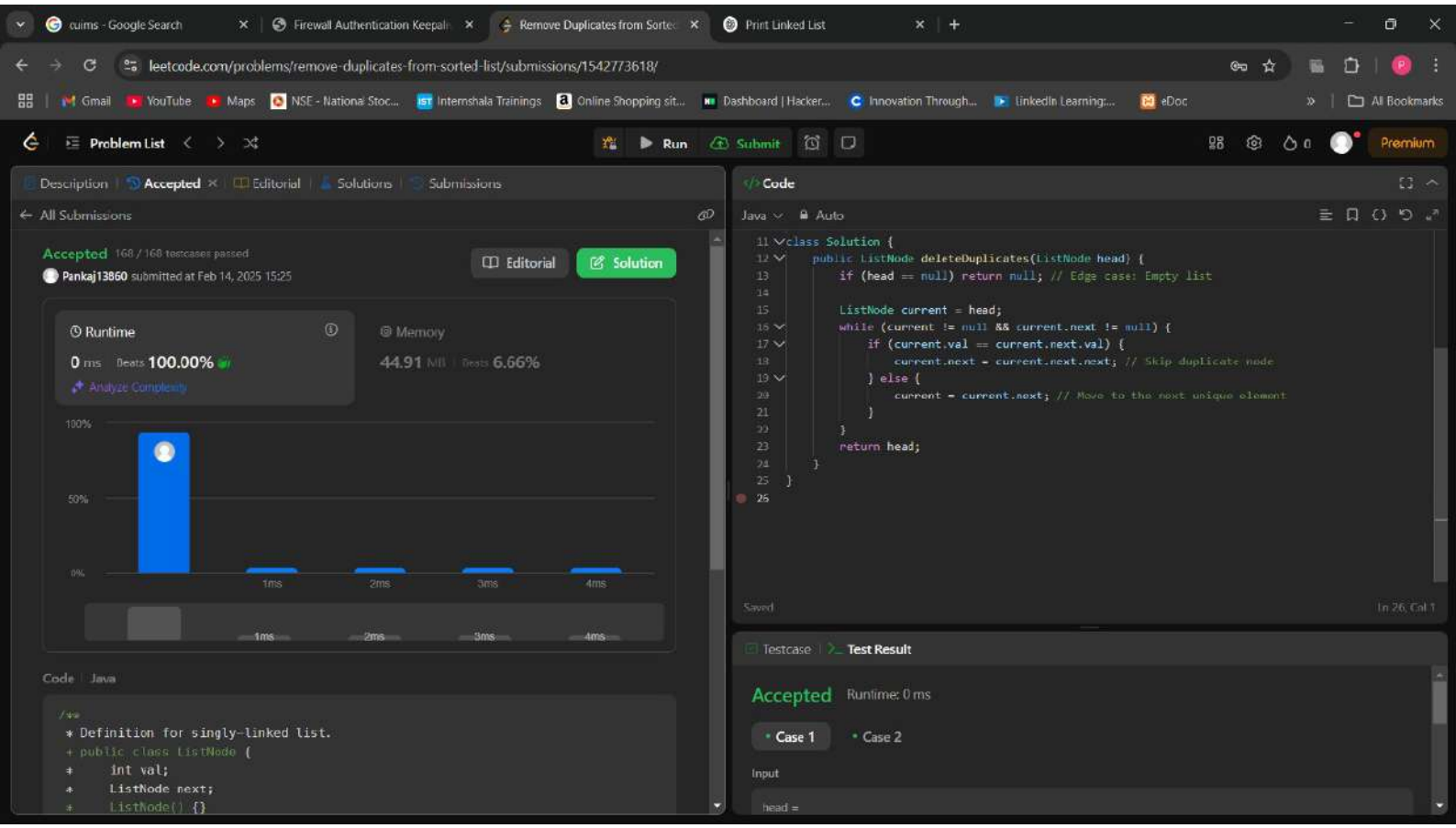
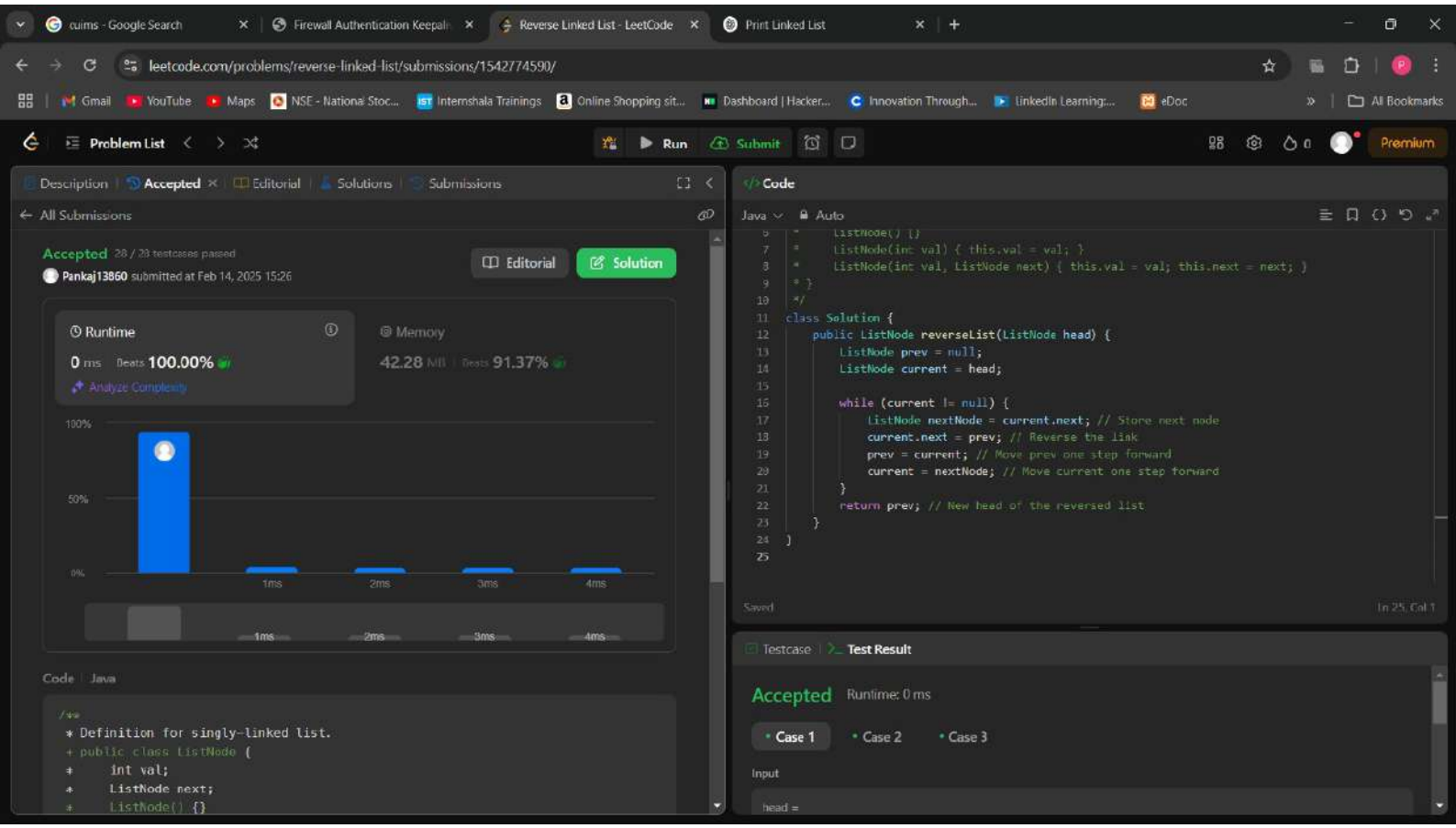
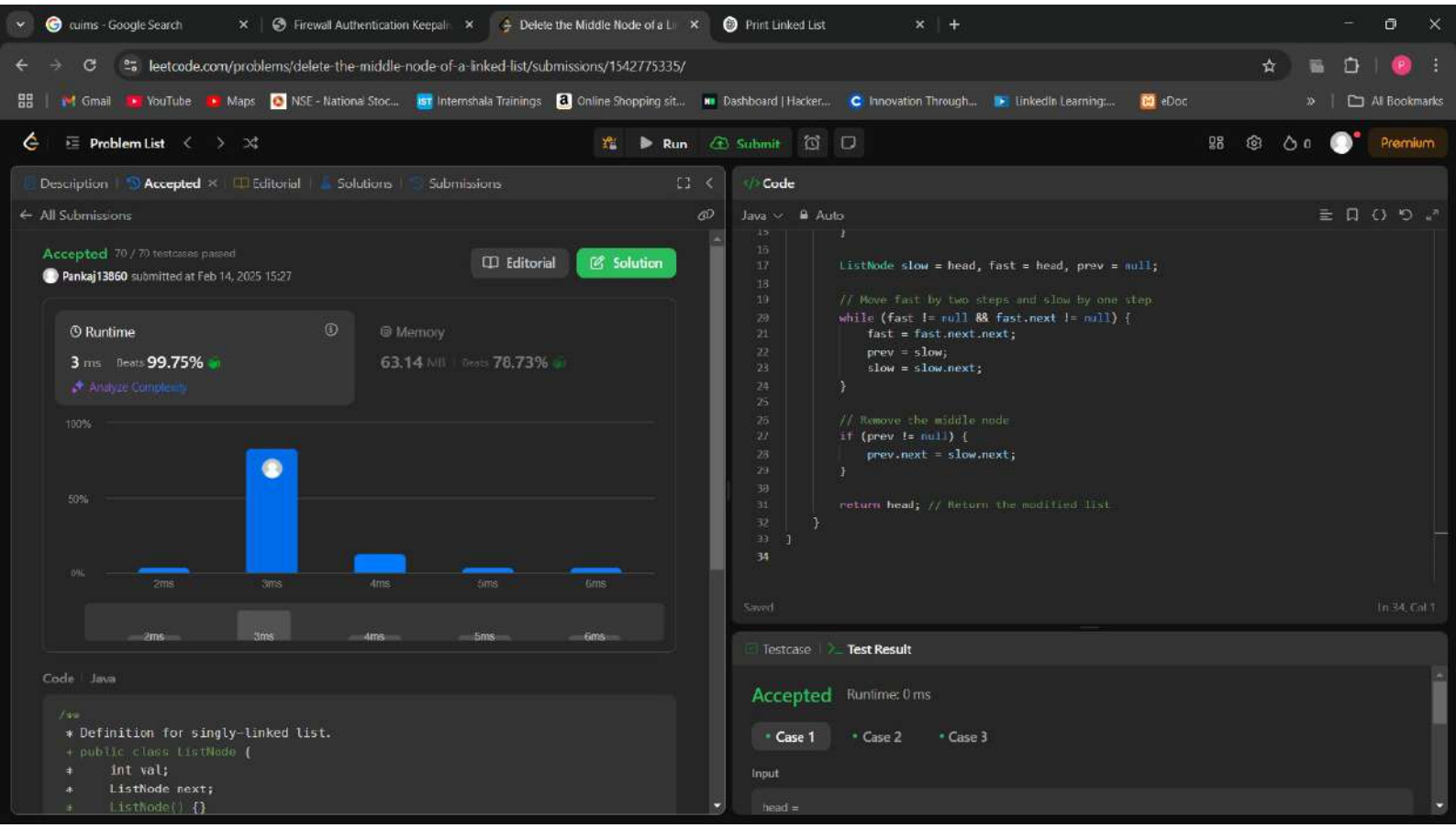
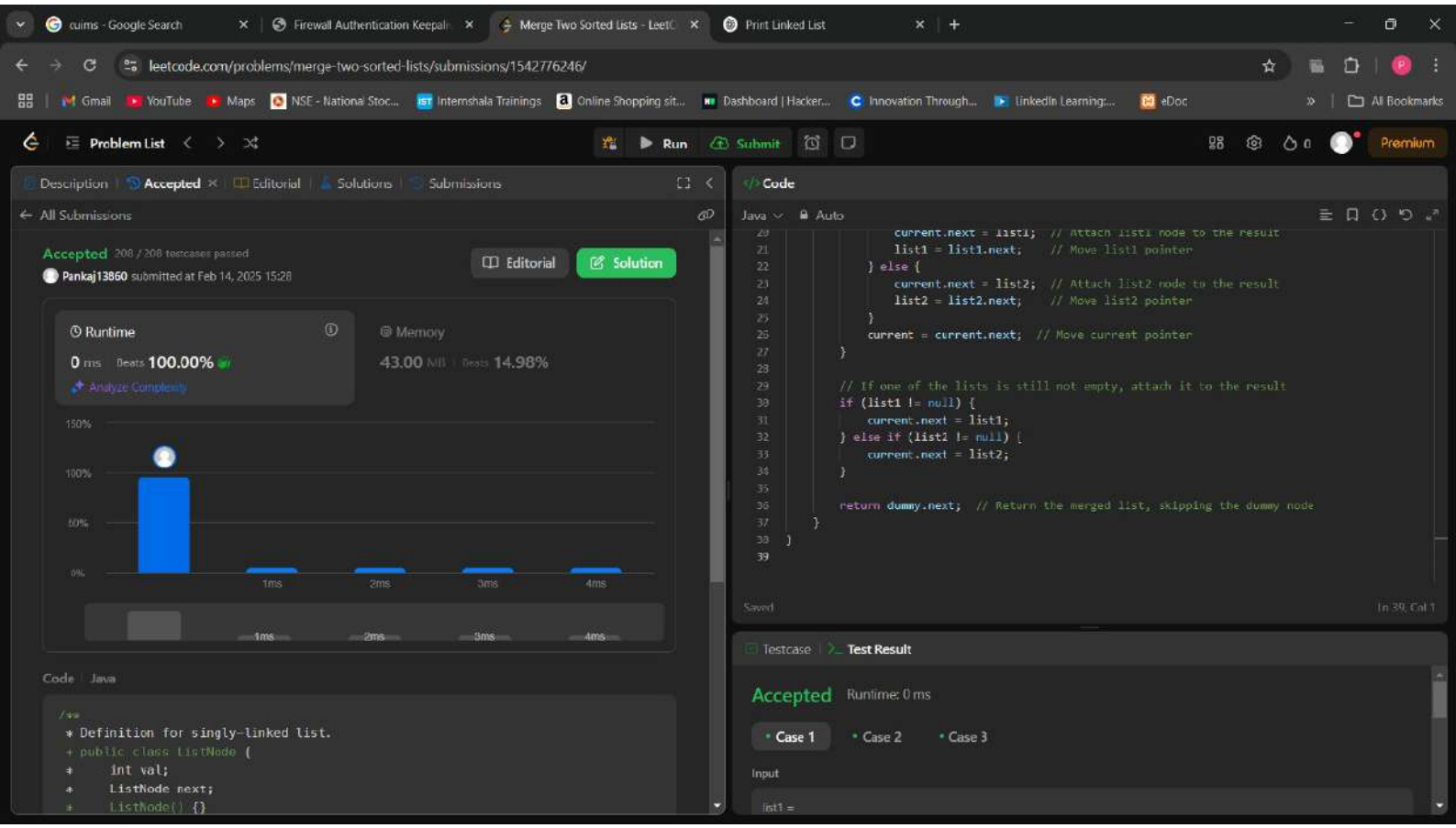
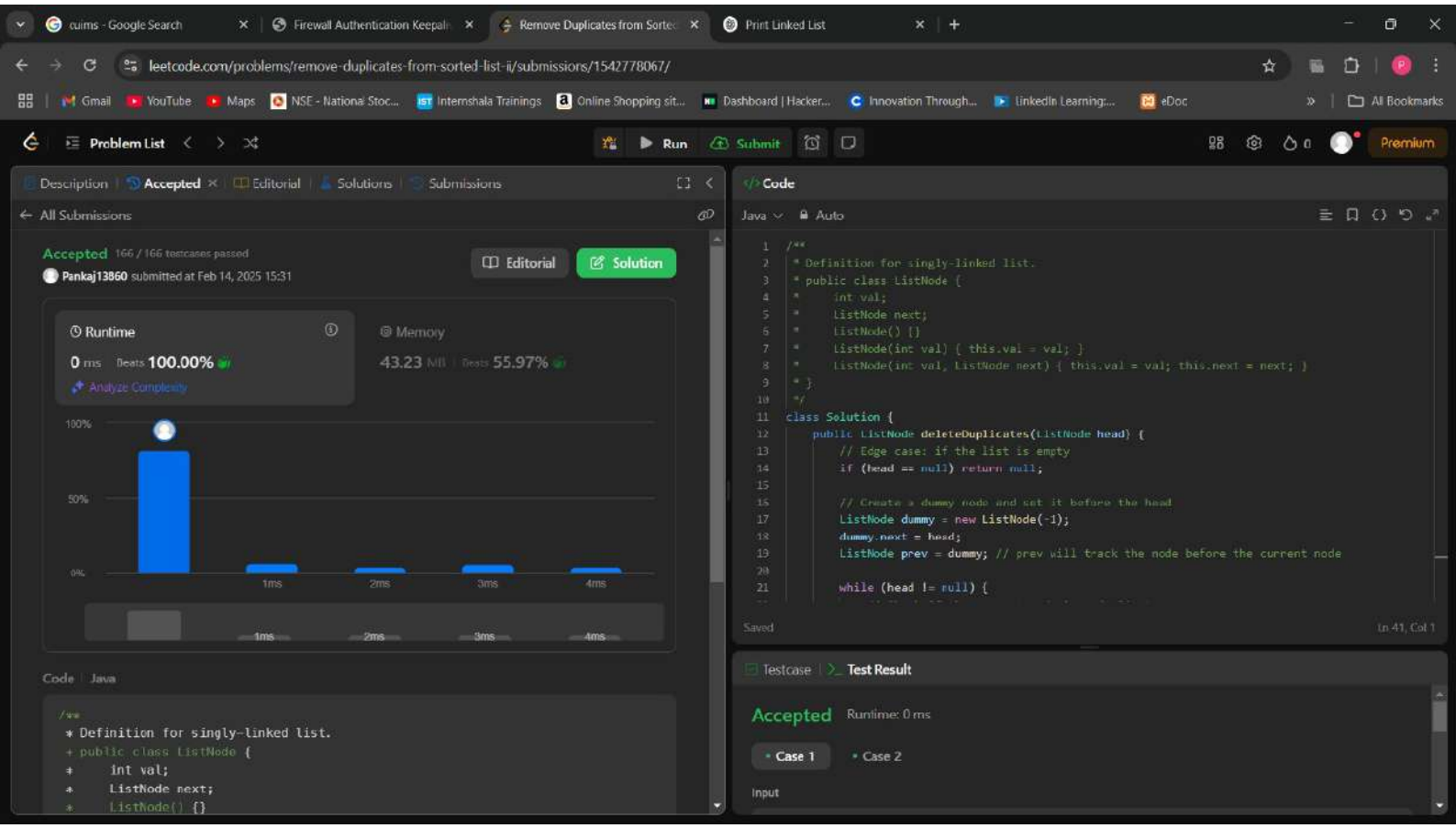

cuims - Google Search
Firewall Authentication Keepalive
Print Linked List | Practice | GeeksforGeeks
Print Linked List
geeksforgeeks.org/problems/print-linked-list-elements/0
Courses
Tutorials
Jobs
Practice
Contests
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1112 / 1112
Attempts : Correct / Total
1 / 3
Accuracy : 33%
Points Scored
1 / 1
Your Total Score: 1
Time Taken
1.79
Solve Next
Count Linked List Nodes
Delete Alternate Nodes
Insert in Middle of Linked List
Kick start your career with GfG 160!
Java (1.8)
Start Timer
// { Driver Code Ends
/* Node is defined as
class Node {
int data;
Node next;
Node(int x) {
data = x;
next = null;
}
}*/
/*
Print elements of a linked list on console
Head pointer input could be NULL as well for empty list
*/
class Solution {
// Function to display the elements of a linked list in same line
void printList(Node head) {
Node temp=head;
while(temp!=null)
{
System.out.print(temp.data+" ");
temp=temp.next;
}
}
}
Custom Input
Compile & Run
Submit

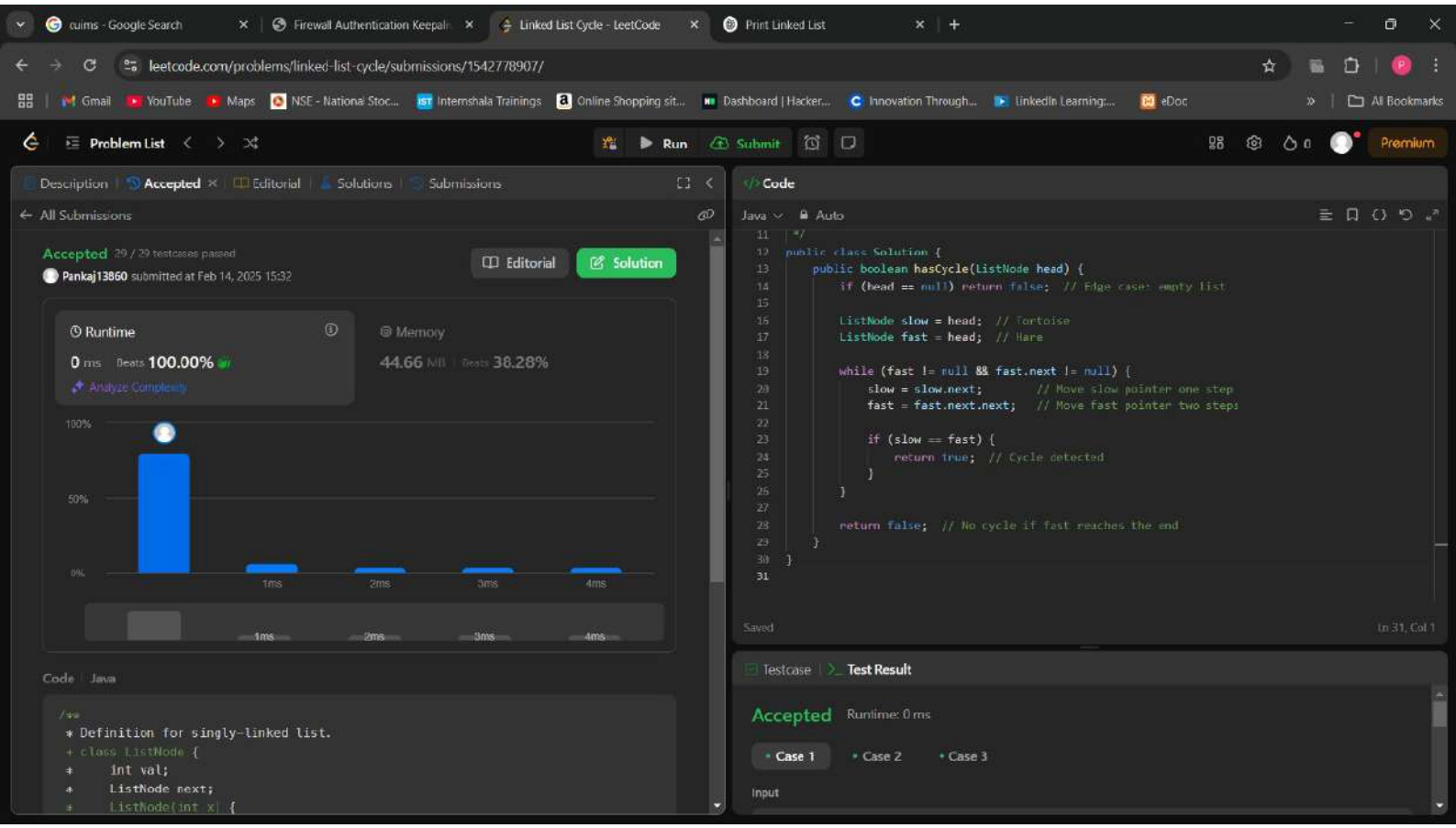

cuims - Google Search
Firewall Authentication Keepali
Remove Duplicates from Sorte
Print Linked List
leetcode.com/problems/remove-duplicates-from-sorted-list/submissions/1542773618/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 168 / 168 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:25
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
44.91 MB Beats 6.66%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
class Solution {
public ListNode deleteDuplicates(ListNode head) {
if (head == null) return null; // Edge case: Empty list
ListNode current = head;
while (current != null && current.next != null) {
if (current.val == current.next.val) {
current.next = current.next.next; // Skip duplicate node
} else {
current = current.next; // Move to the next unique element
}
}
return head;
}
}
Saved
Ln 26, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =

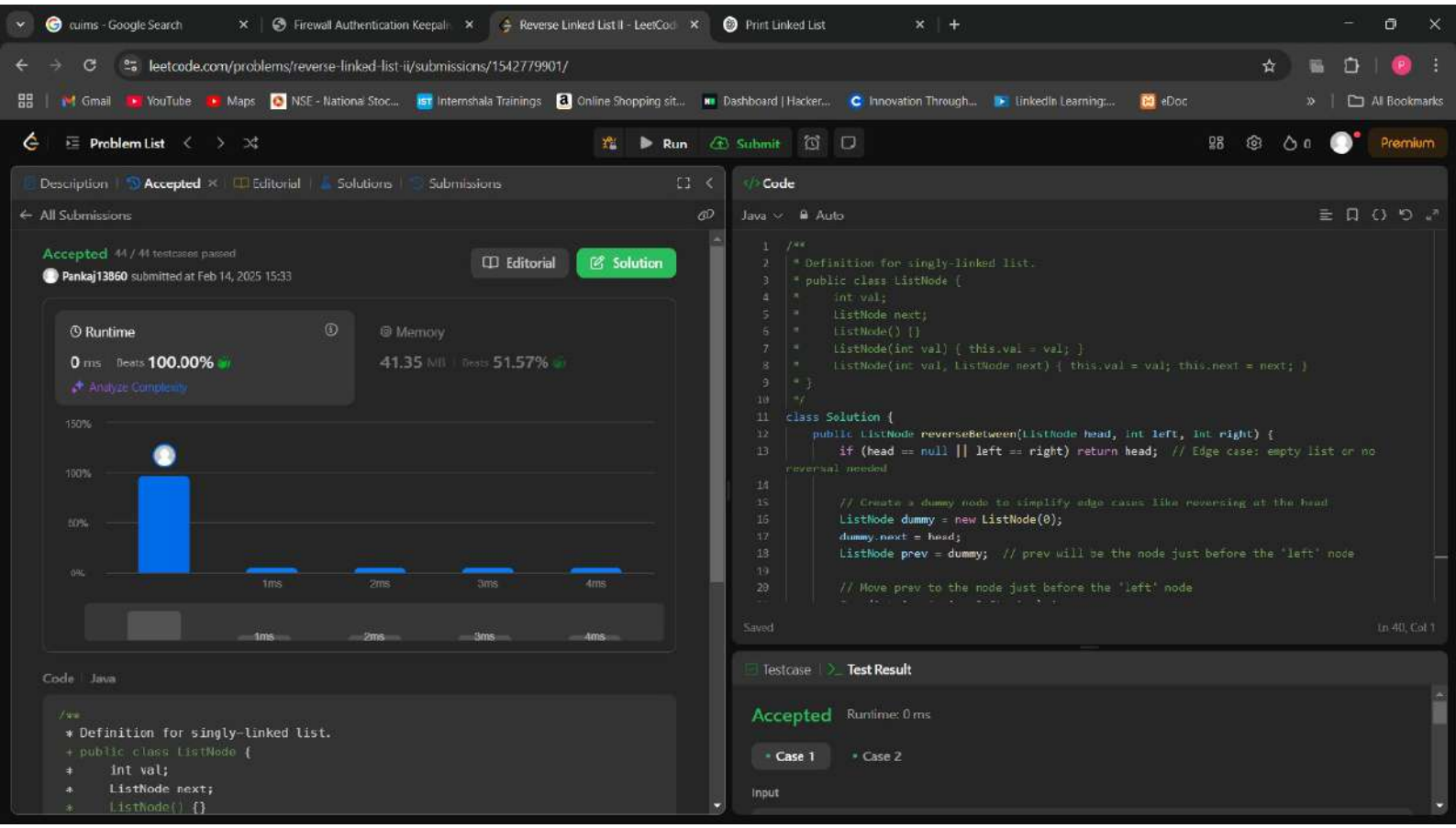

cuims - Google Search
Firewall Authentication Keepali
Reverse Linked List - LeetCode
Print Linked List
leetcode.com/problems/reverse-linked-list/submissions/1542774590/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 28 / 28 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:26
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
42.28 MB Beats 91.37%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
*   ListNode(int val) { this.val = val; }
*   ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;
        while (current != null) {
            ListNode nextNode = current.next; // Store next node
            current.next = prev; // Reverse the link
            prev = current; // Move prev one step forward
            current = nextNode; // Move current one step forward
        }
        return prev; // New head of the reversed list
    }
}
Code
Java
Auto
Saved
Ln 25, Col 1
Testcase | Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =

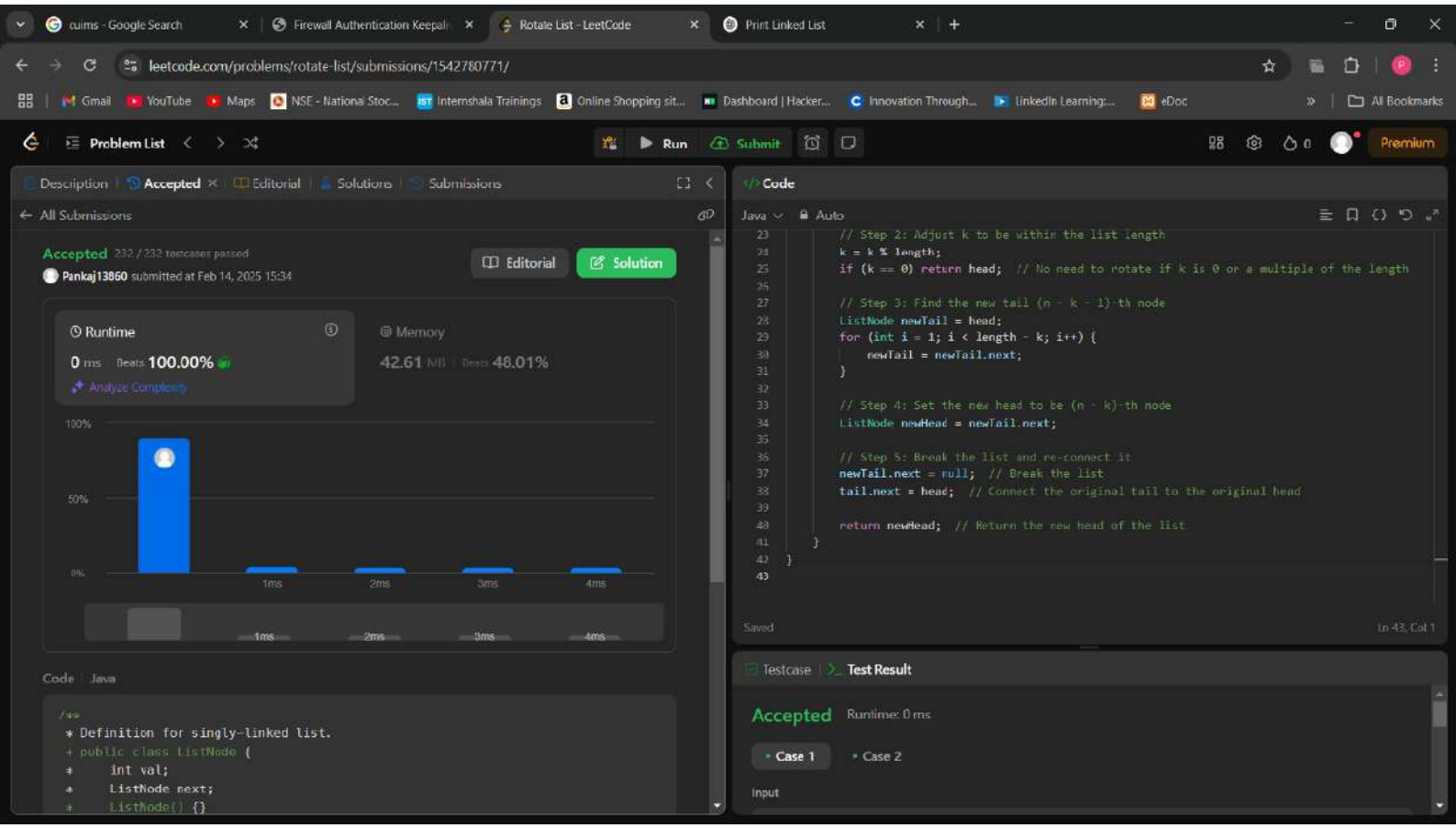

cuims - Google Search
Firewall Authentication Keepalive
Delete the Middle Node of a Li
Print Linked List
leetcode.com/problems/delete-the-middle-node-of-a-linked-list/submissions/1542775335/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 70 / 70 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:27
Editorial
Solution
Runtime
3 ms Beats 99.75%
Analyze Complexity
Memory
63.14 MB Beats 78.73%
100%
50%
0%
2ms
3ms
4ms
5ms
6ms
Code
Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
ListNode slow = head, fast = head, prev = null;
// Move fast by two steps and slow by one step
while (fast != null && fast.next != null) {
fast = fast.next.next;
prev = slow;
slow = slow.next;
}
// Remove the middle node
if (prev != null) {
prev.next = slow.next;
}
return head; // Return the modified list
}
}
Saved
Ln 34, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =

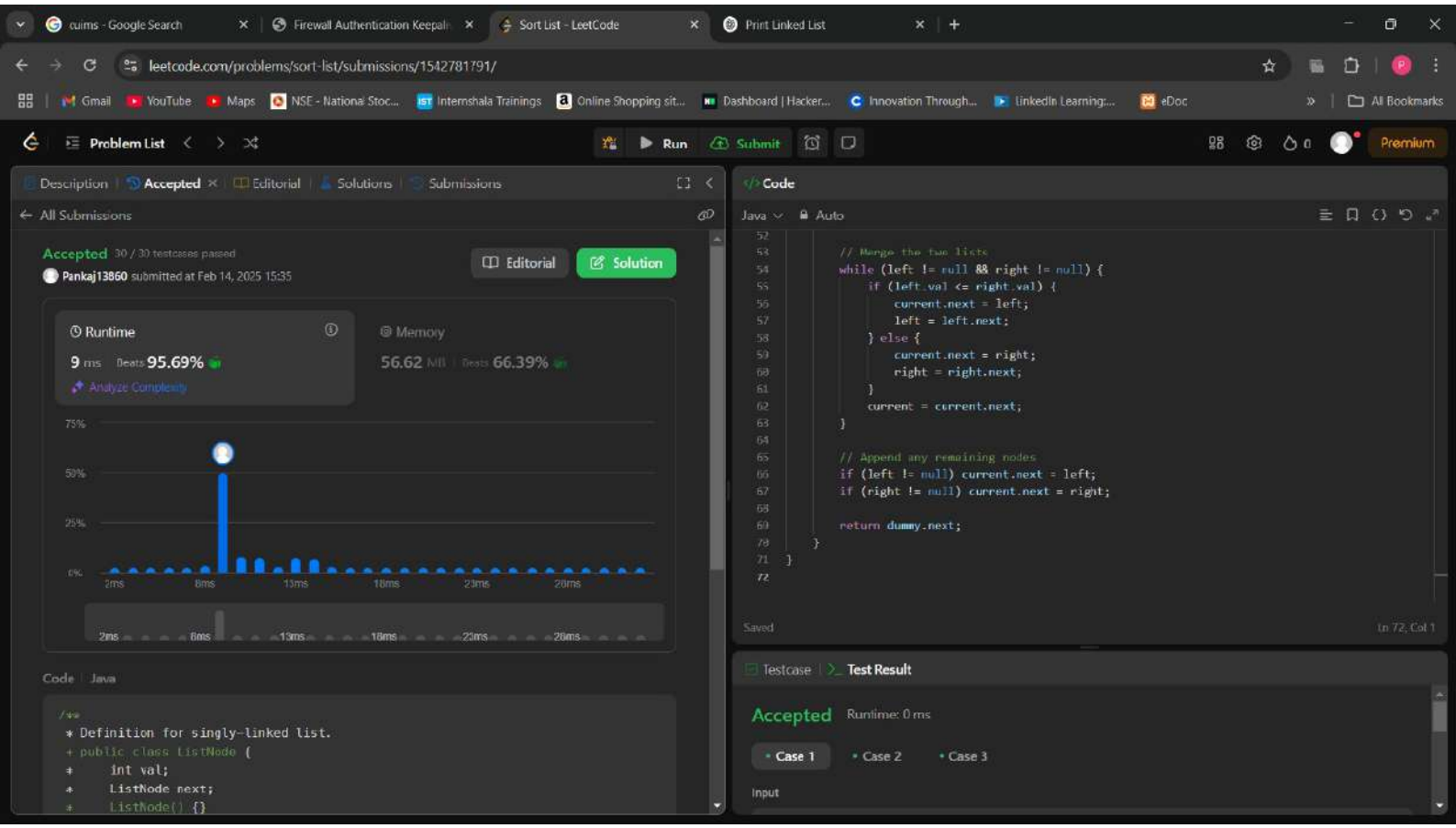

cuims - Google Search
Firewall Authentication Keepali
Merge Two Sorted Lists - LeetC
Print Linked List
leetcode.com/problems/merge-two-sorted-lists/submissions/1542776246/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 208 / 208 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:28
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
43.00 MB Beats 14.98%
150%
100%
50%
0%
1ms
2ms
3ms
4ms
Code
Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
current.next = list1; // Attach list1 node to the result
list1 = list1.next; // Move list1 pointer
} else {
current.next = list2; // Attach list2 node to the result
list2 = list2.next; // Move list2 pointer
}
current = current.next; // Move current pointer
}
// If one of the lists is still not empty, attach it to the result
if (list1 != null) {
current.next = list1;
} else if (list2 != null) {
current.next = list2;
}
return dummy.next; // Return the merged list, skipping the dummy node
}
}
Saved
Ln 39, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
list1 =

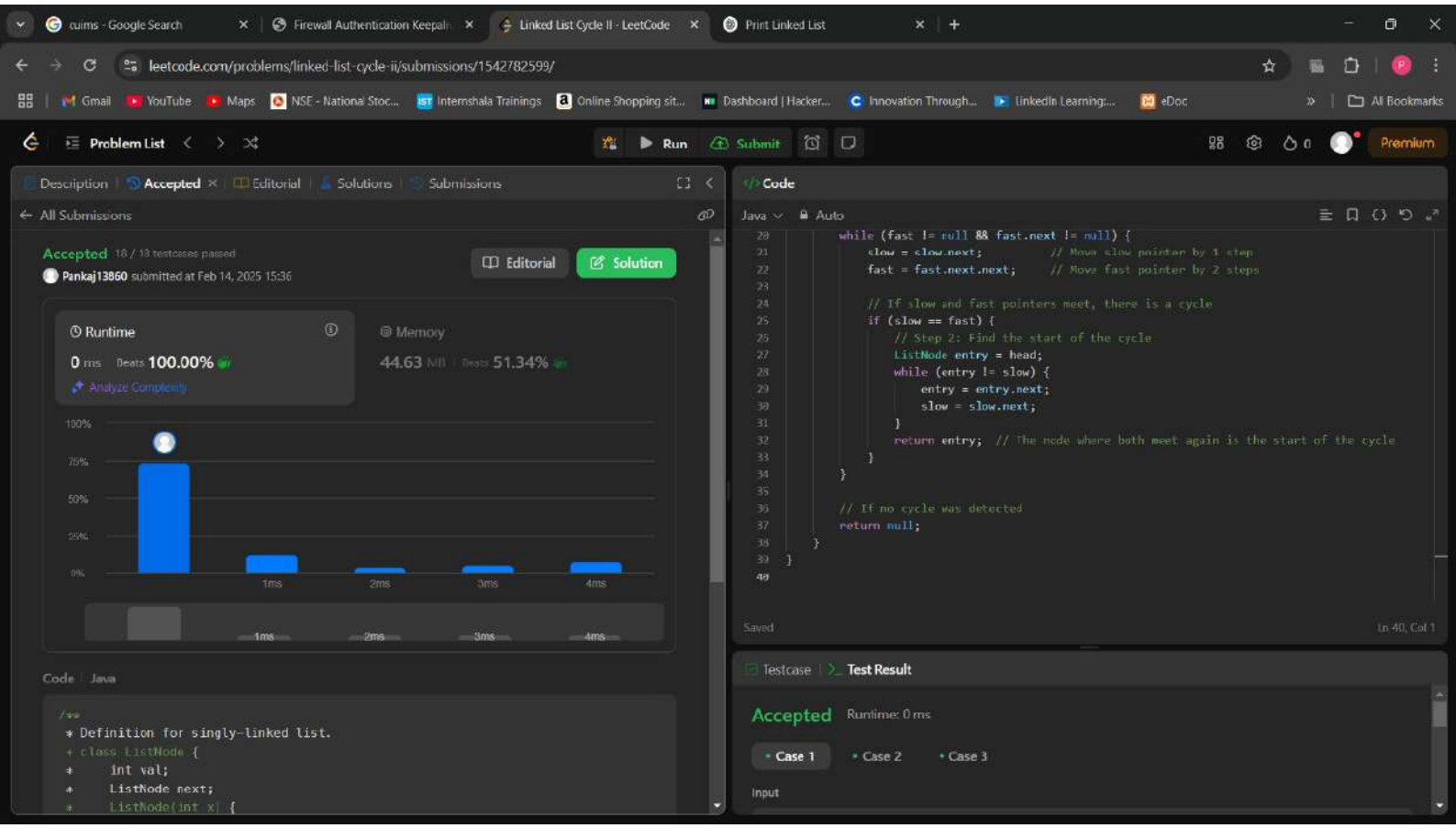

cuims - Google Search
Firewall Authentication Keepali
Remove Duplicates from Sorte
Print Linked List
leetcode.com/problems/remove-duplicates-from-sorted-list-ii/submissions/1542778067/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 166 / 166 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:31
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
43.23 MB Beats 55.97%
100%
50%
0%
1ms
2ms
3ms
4ms
Code
Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() []
*   ListNode(int val) { this.val = val; }
*   ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
public ListNode deleteDuplicates(ListNode head) {
// Edge case: if the list is empty
if (head == null) return null;
// Create a dummy node and set it before the head
ListNode dummy = new ListNode(-1);
dummy.next = head;
ListNode prev = dummy; // prev will track the node before the current node
while (head != null) {
Saved
Ln 41, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input


cuims - Google Search
Firewall Authentication Keepalive
Linked List Cycle - LeetCode
Print Linked List
leetcode.com/problems/linked-list-cycle/submissions/1542778907/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 29 / 29 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:32
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
44.66 MB Beats 38.28%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | Java
/**
* Definition for singly-linked list.
* class ListNode {
*   int val;
*   ListNode next;
*   ListNode(int x) {
Code
Java
Auto
*/
public class Solution {
public boolean hasCycle(ListNode head) {
if (head == null) return false; // Edge case: empty list
ListNode slow = head; // Tortoise
ListNode fast = head; // Hare
while (fast != null && fast.next != null) {
slow = slow.next; // Move slow pointer one step
fast = fast.next.next; // Move fast pointer two steps
if (slow == fast) {
return true; // Cycle detected
}
}
return false; // No cycle if fast reaches the end
}
}
Saved
Ln 31, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input


cuims - Google Search
Firewall Authentication Keepalive
Reverse Linked List II - LeetCode
Print Linked List
leetcode.com/problems/reverse-linked-list-ii/submissions/1542779901/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 44 / 44 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:33
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
41.35 MB Beats 51.57%
150%
100%
50%
0%
1ms
2ms
3ms
4ms
Code
Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() []
*   ListNode(int val) { this.val = val; }
*   ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
public ListNode reverseBetween(ListNode head, int left, int right) {
if (head == null || left == right) return head; // Edge case: empty list or no reversal needed
// Create a dummy node to simplify edge cases like reversing at the head
ListNode dummy = new ListNode(0);
dummy.next = head;
ListNode prev = dummy; // prev will be the node just before the 'left' node
// Move prev to the node just before the 'left' node
Saved
Ln 40, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input


cuims - Google Search
Firewall Authentication Keepali
Rotate List - LeetCode
Print Linked List
leetcode.com/problems/rotate-list/submissions/1542780771/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 232 / 232 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:34
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
42.61 MB Beats 48.01%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
// Step 2: Adjust k to be within the list length
k = k % length;
if (k == 0) return head; // No need to rotate if k is 0 or a multiple of the length
// Step 3: Find the new tail (n - k - 1)-th node
ListNode newTail = head;
for (int i = 1; i < length - k; i++) {
newTail = newTail.next;
}
// Step 4: Set the new head to be (n - k)-th node
ListNode newHead = newTail.next;
// Step 5: Break the list and re-connect it
newTail.next = null; // Break the list
tail.next = head; // Connect the original tail to the original head
return newHead; // Return the new head of the list
}
}
Saved
Ln 43, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input


cuims - Google Search
Firewall Authentication Keepali
Sort List - LeetCode
Print Linked List
leetcode.com/problems/sort-list/submissions/1542781791/
Gmail
YouTube
Maps
NSE - National Stoc...
Internshala Trainings
Online Shopping sit...
Dashboard | Hacker...
Innovation Through...
LinkedIn Learning:...
eDoc
All Bookmarks
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 30 / 30 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:35
Editorial
Solution
Runtime
9 ms Beats 95.69%
Analyze Complexity
Memory
56.62 MB Beats 66.39%
75%
50%
25%
0%
2ms
6ms
10ms
18ms
23ms
26ms
Code
Java
/**
* Definition for singly-linked list.
* public class ListNode {
*   int val;
*   ListNode next;
*   ListNode() {}
Code
Java
Auto
// Merge the two lists
while (left != null && right != null) {
if (left.val <= right.val) {
current.next = left;
left = left.next;
} else {
current.next = right;
right = right.next;
}
current = current.next;
}
// Append any remaining nodes
if (left != null) current.next = left;
if (right != null) current.next = right;
return dummy.next;
}
}
Saved
Ln 72, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input


cuims - Google Search
Firewall Authentication Keepalive
Linked List Cycle II - LeetCode
Print Linked List
leetcode.com/problems/linked-list-cycle-ii/submissions/1542782599/
Problem List
Run
Submit
Premium
Description | Accepted | Editorial | Solutions | Submissions
All Submissions
Accepted 18 / 18 testcases passed
Pankaj13860 submitted at Feb 14, 2025 15:36
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
44.63 MB Beats 51.34%
100%
75%
50%
25%
0%
1ms
2ms
3ms
4ms
Code | Java
/**
* Definition for singly-linked list.
* class ListNode {
*   int val;
*   ListNode next;
*   ListNode(int x) {
Code
Java
Auto
while (fast != null && fast.next != null) {
slow = slow.next; // Move slow pointer by 1 step
fast = fast.next.next; // Move fast pointer by 2 steps
// If slow and fast pointers meet, there is a cycle
if (slow == fast) {
// Step 2: Find the start of the cycle
ListNode entry = head;
while (entry != slow) {
entry = entry.next;
slow = slow.next;
}
return entry; // The node where both meet again is the start of the cycle
}
}
// If no cycle was detected
return null;
}
}
Saved
Ln 40, Col 1
Testcase | Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input