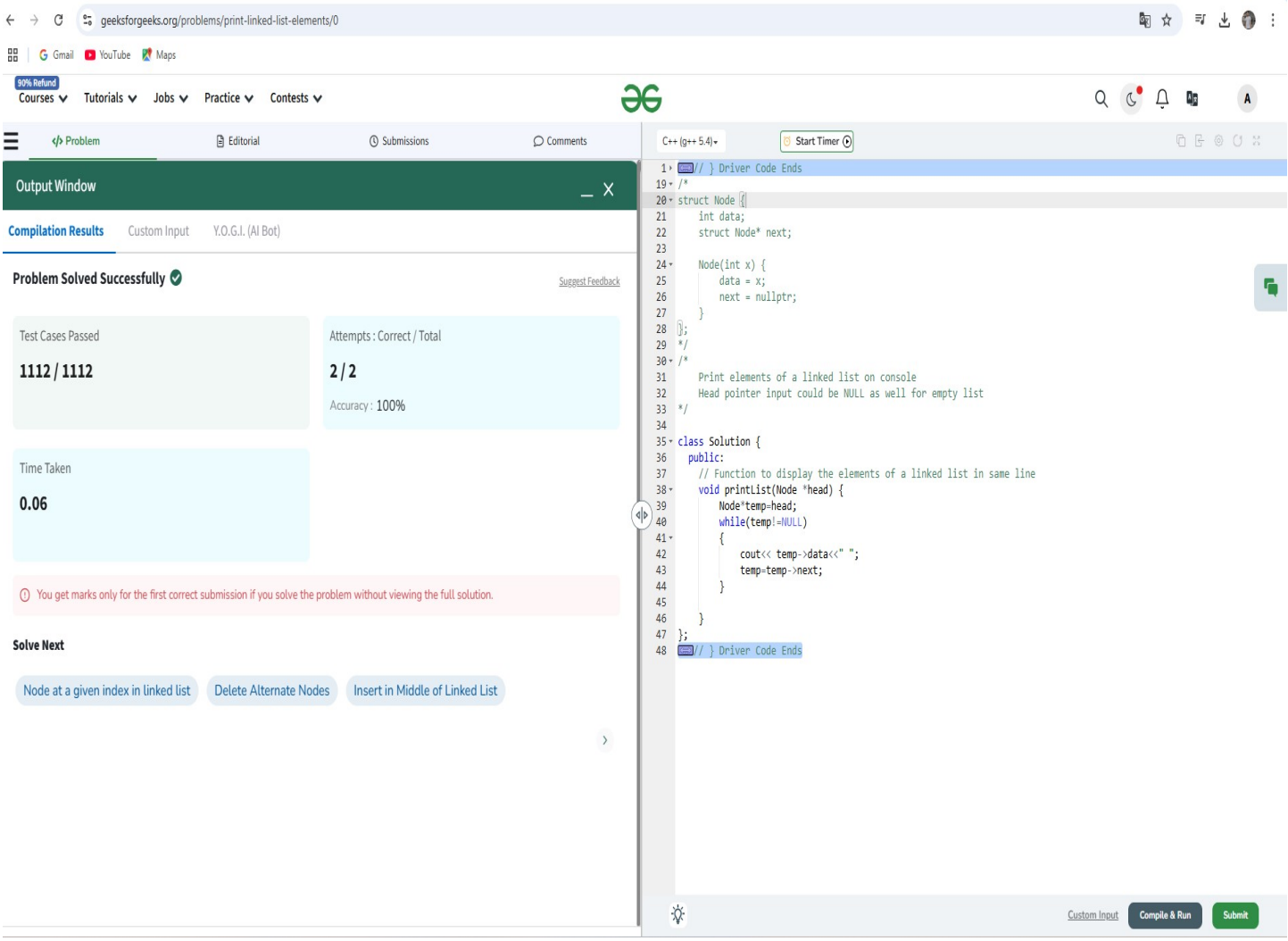
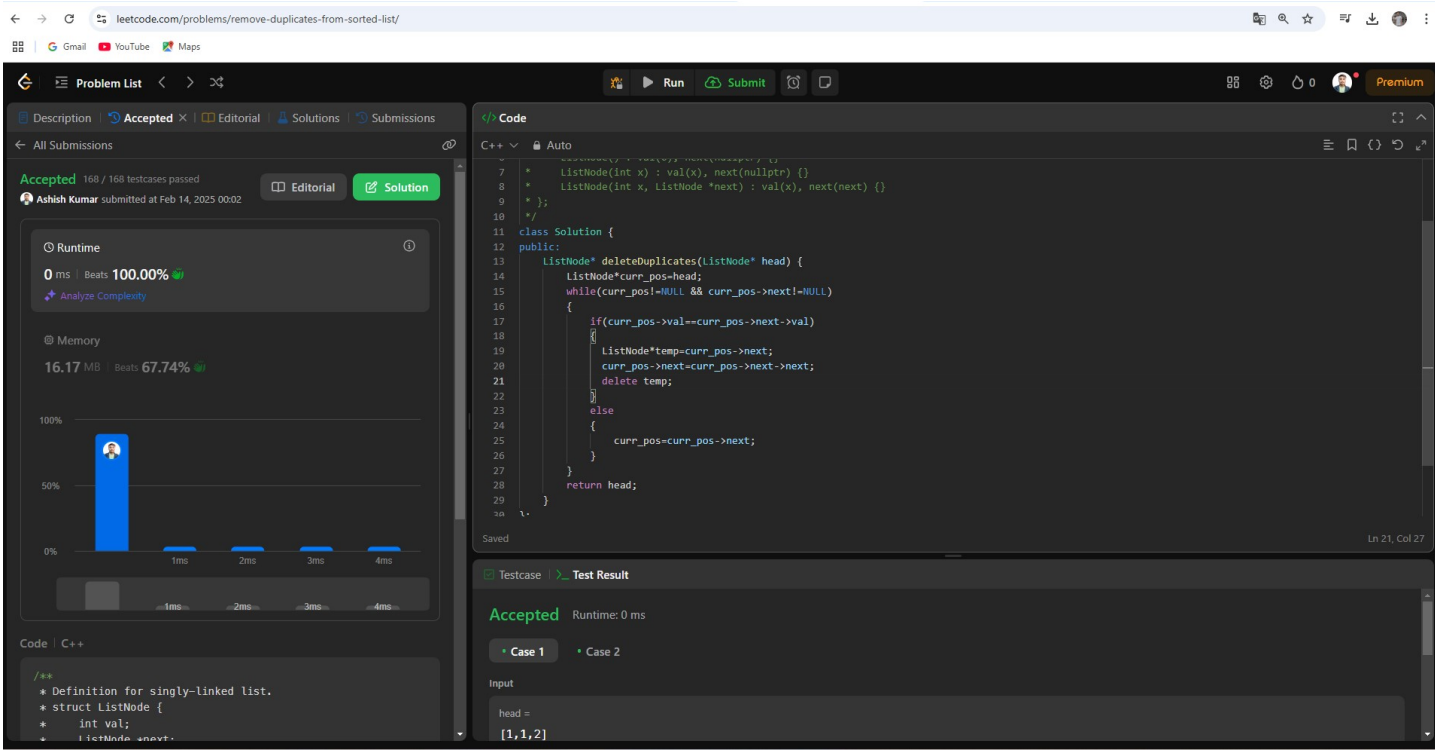
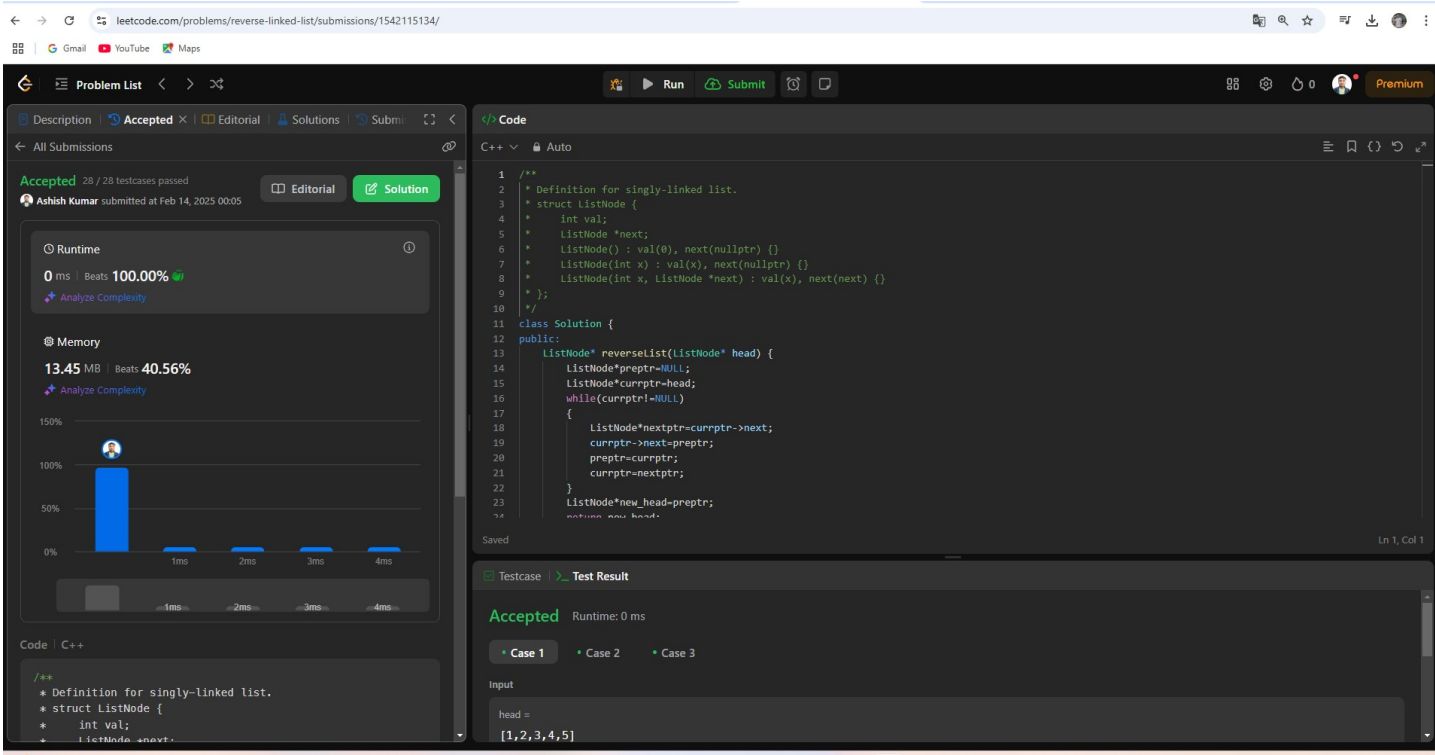
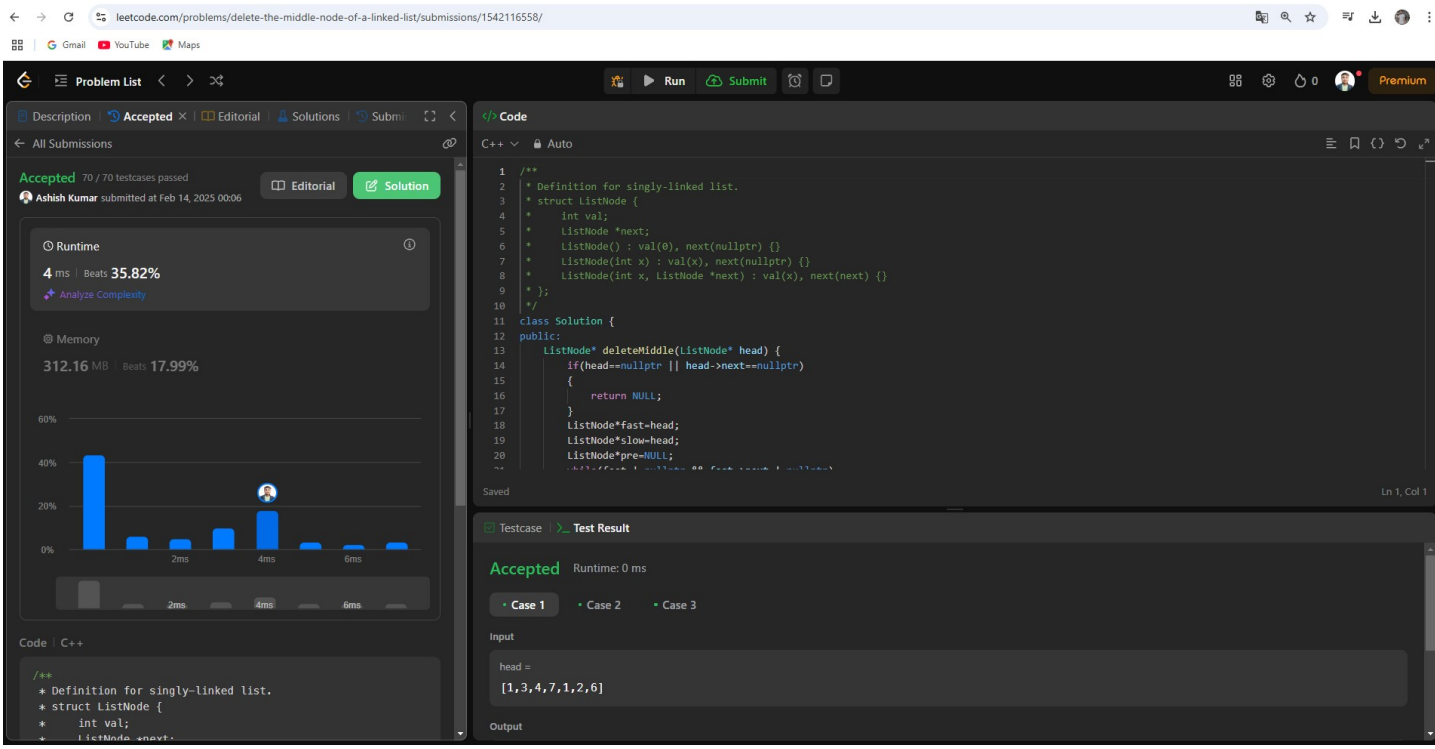
Experiment 3

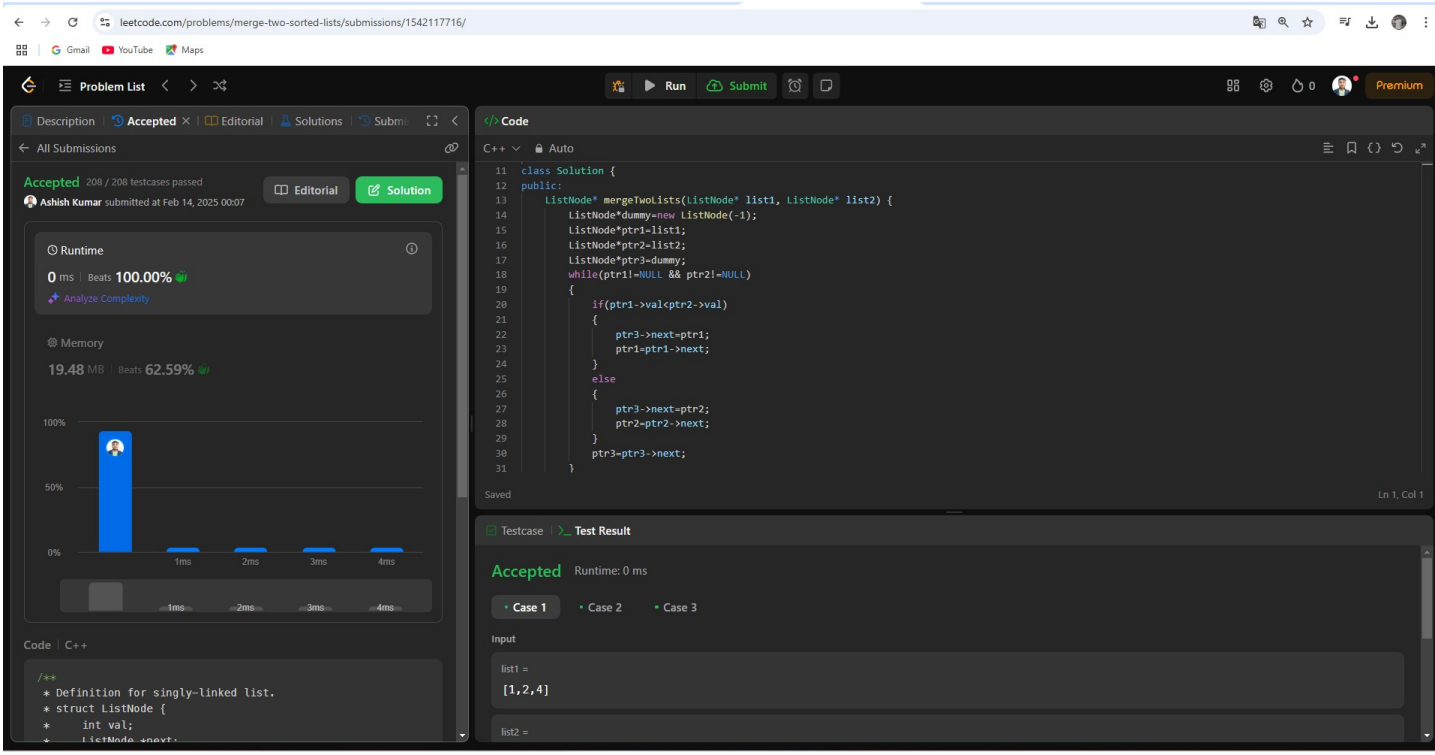
Student Name: Ashish Kumar **UID: 22BCS11958**
Branch: CSE **Section/Group: 614/B**
Semester: 6 **Date of Performance: 15/02/2025**
Subject Name: AP lab-2 **Subject Code: 22CSP-351**

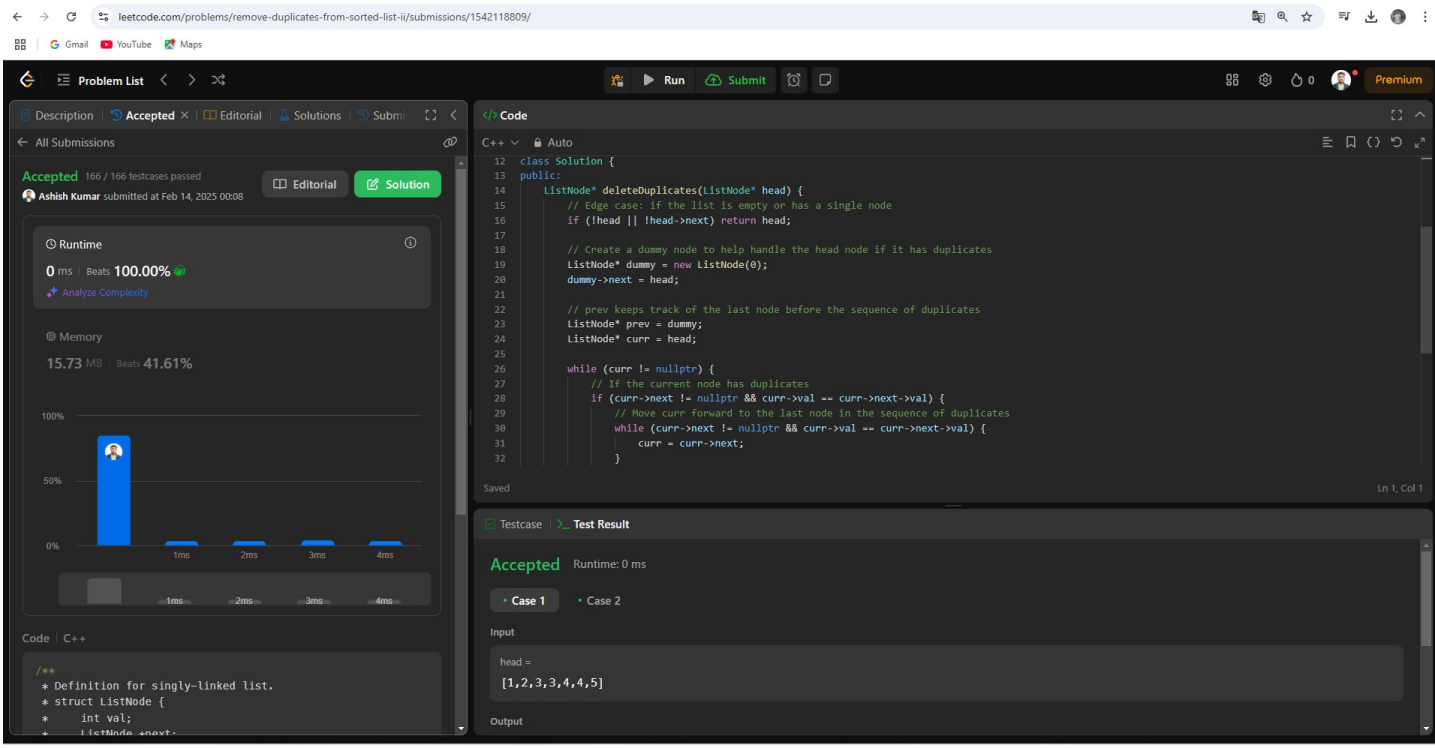


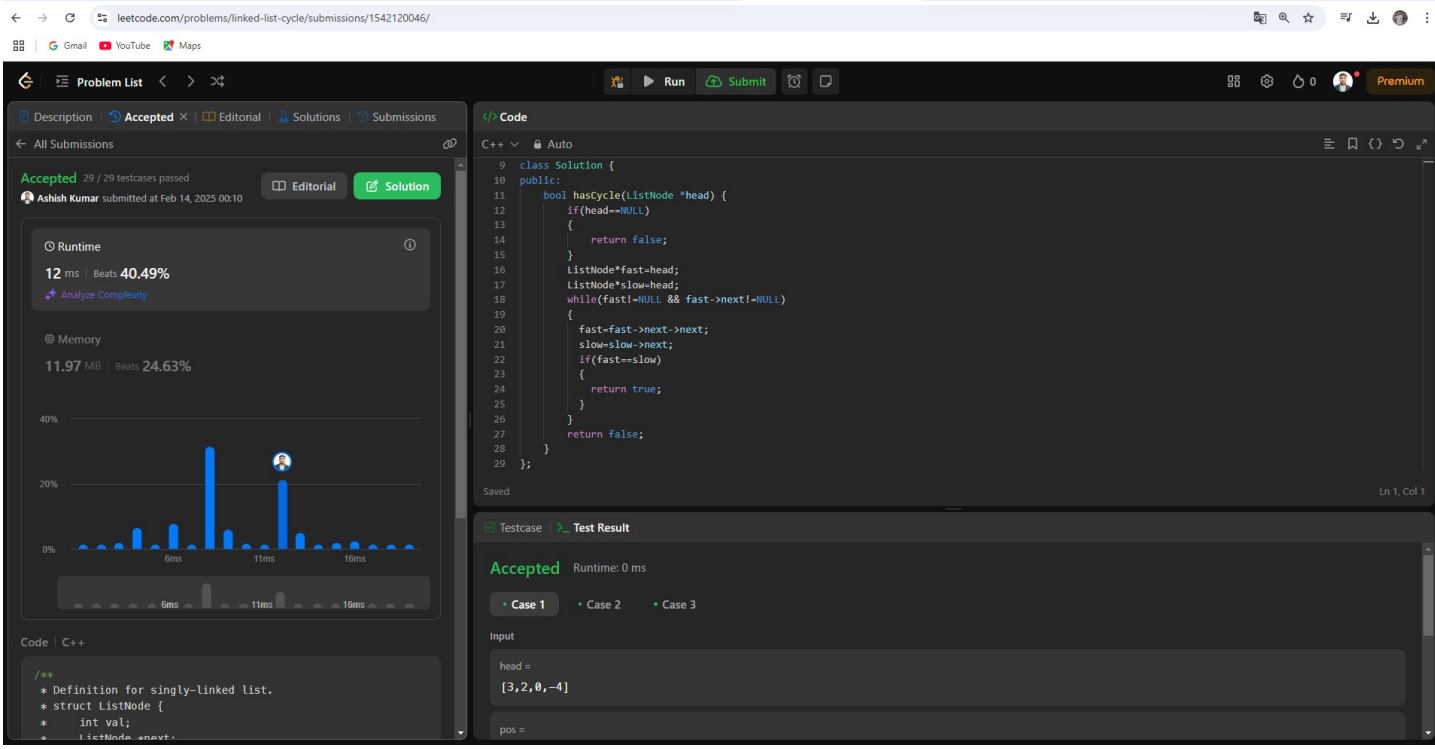

leetcode.com/problems/remove-duplicates-from-sorted-list/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 168 / 168 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:02
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
16.17 MB | Beats 67.74%
Code
C++
Auto
class Solution {
public:
ListNode* deleteDuplicates(ListNode* head) {
ListNode*curr_pos=head;
while(curr_pos!=NULL && curr_pos->next!=NULL)
if(curr_pos->val==curr_pos->next->val)
ListNode*temp=curr_pos->next;
curr_pos->next=curr_pos->next->next;
delete temp;
else
curr_pos=curr_pos->next;
return head;
Saved
Ln 21, Col 27
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,1,2]

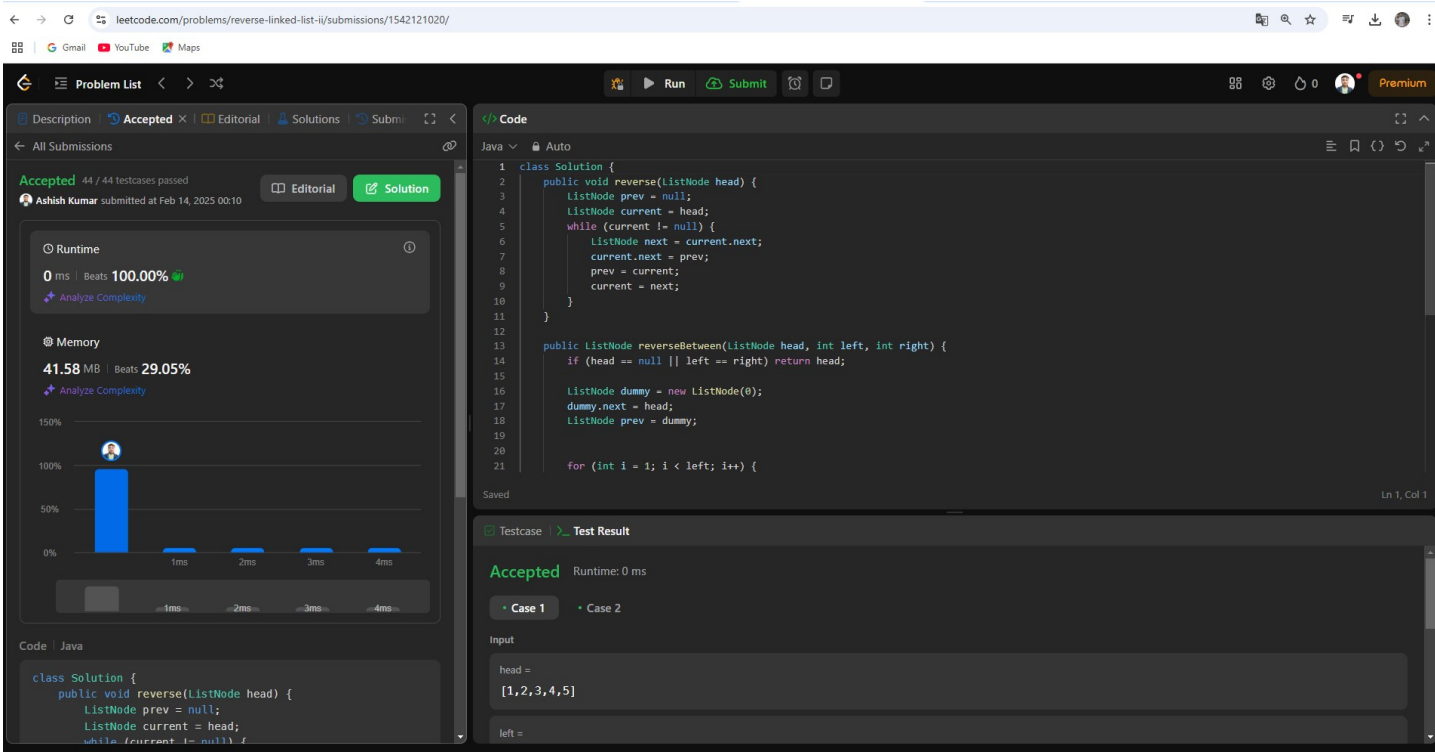

leetcode.com/problems/reverse-linked-list/submissions/1542115134/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 28 / 28 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:05
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
13.45 MB | Beats 40.56%
Analyze Complexity
Code
C++
Auto
/**
* 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* reverseList(ListNode* head) {
ListNode*preptr=NULL;
ListNode*currptr=head;
while(currptr!=NULL)
ListNode*nextptr=currptr->next;
currptr->next=preptr;
preptr=currptr;
currptr=nextptr;
ListNode*new_head=preptr;
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[1,2,3,4,5]

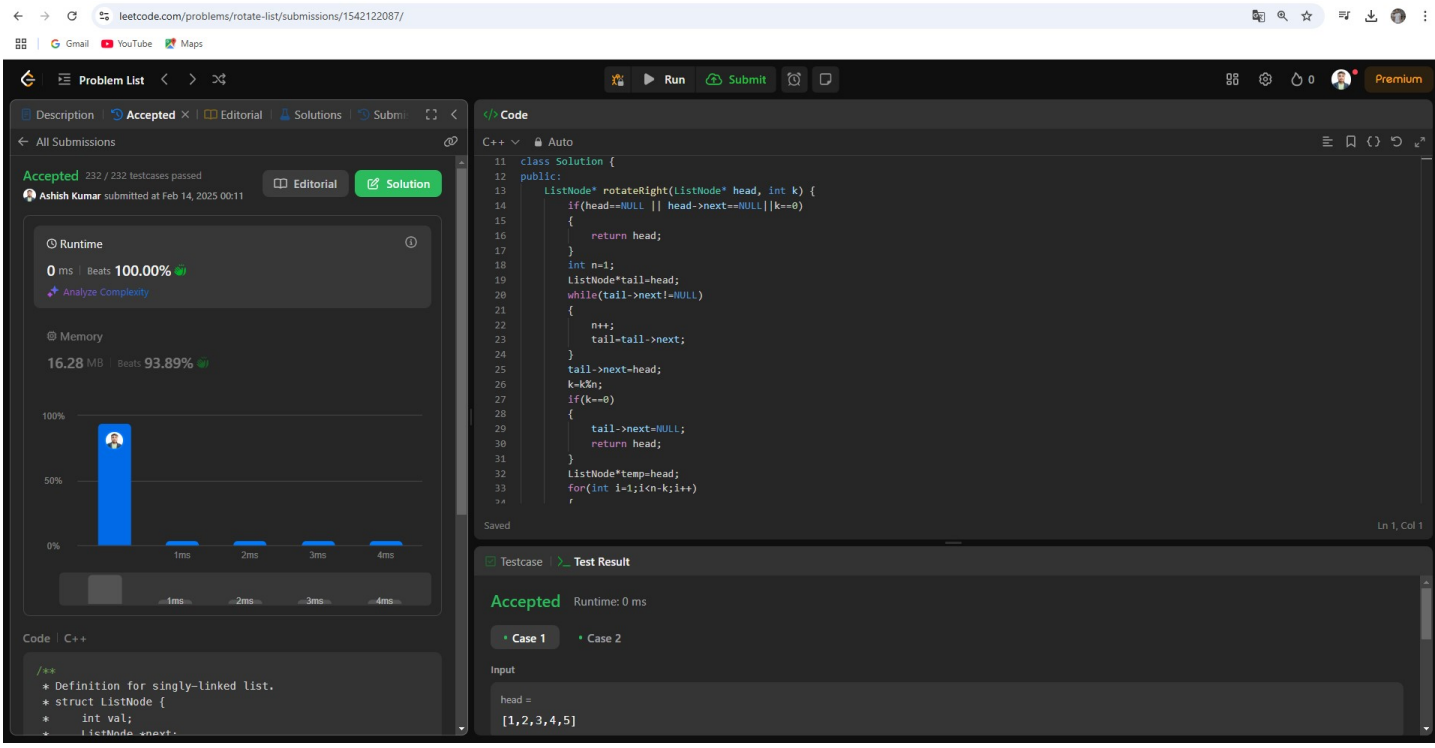

leetcode.com/problems/delete-the-middle-node-of-a-linked-list/submissions/1542116558/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 70 / 70 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:06
Editorial
Solution
Runtime
4 ms | Beats 35.82%
Analyze Complexity
Memory
312.16 MB | Beats 17.99%
Code
C++
Auto
/**
* 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==nullptr || head->next==nullptr)
{
return NULL;
}
ListNode*fast=head;
ListNode*slow=head;
ListNode*pre=NULL;
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[1,3,4,7,1,2,6]
Output

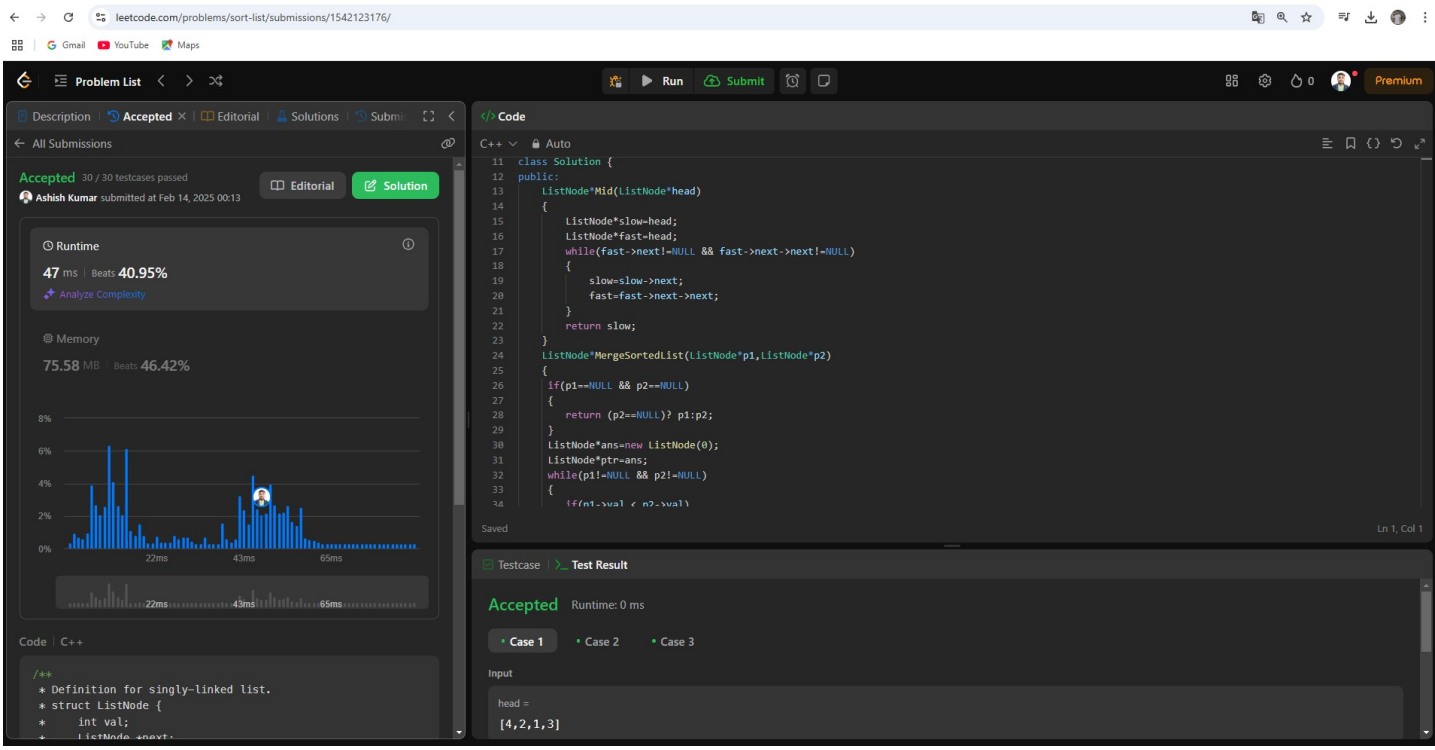

leetcode.com/problems/merge-two-sorted-lists/submissions/1542117716/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 208 / 208 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:07
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
19.48 MB | Beats 62.59%
Code
C++
Auto
class Solution {
public:
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
ListNode*dummy=new ListNode(-1);
ListNode*ptr1=list1;
ListNode*ptr2=list2;
ListNode*ptr3=dummy;
while(ptr1!=NULL && ptr2!=NULL)
{
if(ptr1->val<ptr2->val)
{
ptr3->next=ptr1;
ptr1=ptr1->next;
}
else
{
ptr3->next=ptr2;
ptr2=ptr2->next;
}
ptr3=ptr3->next;
}
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
list1 =
[1,2,4]
list2 =

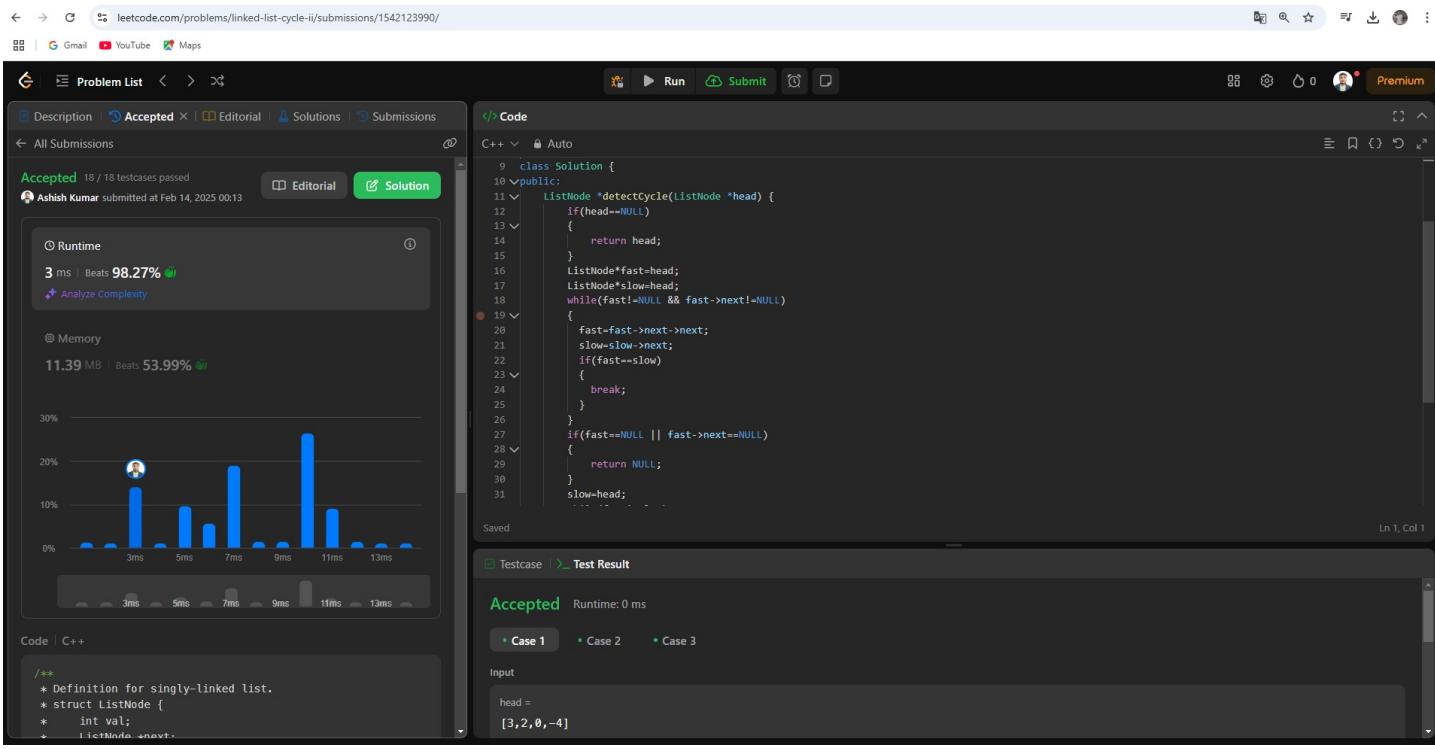

leetcode.com/problems/remove-duplicates-from-sorted-list-ii/submissions/1542118809/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 166 / 166 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:08
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
15.73 MB | Beats 41.61%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
Code
C++
Auto
class Solution {
public:
ListNode* deleteDuplicates(ListNode* head) {
// Edge case: if the list is empty or has a single node
if (!head || !head->next) return head;
// Create a dummy node to help handle the head node if it has duplicates
ListNode* dummy = new ListNode(0);
dummy->next = head;
// prev keeps track of the last node before the sequence of duplicates
ListNode* prev = dummy;
ListNode* curr = head;
while (curr != nullptr) {
// If the current node has duplicates
if (curr->next != nullptr && curr->val == curr->next->val) {
// Move curr forward to the last node in the sequence of duplicates
while (curr->next != nullptr && curr->val == curr->next->val) {
curr = curr->next;
}
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,3,4,4,5]
Output


leetcode.com/problems/linked-list-cycle/submissions/1542120046/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 29 / 29 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:10
Editorial
Solution
Runtime
12 ms | Beats 40.49%
Analyze Complexity
Memory
11.97 MB | Beats 24.63%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
Code
C++
Auto
class Solution {
public:
bool hasCycle(ListNode *head) {
if(head==NULL)
{
return false;
}
ListNode*fast=head;
ListNode*slow=head;
while(fast!=NULL && fast->next!=NULL)
{
fast=fast->next->next;
slow=slow->next;
if(fast==slow)
{
return true;
}
}
return false;
}
};
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[3,2,0,-4]
pos =


leetcode.com/problems/reverse-linked-list-ii/submissions/1542121020/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 44 / 44 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:10
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
41.58 MB | Beats 29.05%
Analyze Complexity
Code
Java
Auto
class Solution {
public void reverse(ListNode head) {
ListNode prev = null;
ListNode current = head;
while (current != null) {
ListNode next = current.next;
current.next = prev;
prev = current;
current = next;
}
}
public ListNode reverseBetween(ListNode head, int left, int right) {
if (head == null || left == right) return head;
ListNode dummy = new ListNode(0);
dummy.next = head;
ListNode prev = dummy;
for (int i = 1; i < left; i++) {
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,4,5]
left =


leetcode.com/problems/rotate-list/submissions/1542122087/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 232 / 232 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:11
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
16.28 MB | Beats 93.89%
Code
C++
Auto
class Solution {
public:
ListNode* rotateRight(ListNode* head, int k) {
if(head==NULL || head->next==NULL||k==0)
{
return head;
}
int n=1;
ListNode*tail=head;
while(tail->next!=NULL)
{
n++;
tail=tail->next;
}
tail->next=head;
k=k%n;
if(k==0)
{
tail->next=NULL;
return head;
}
ListNode*temp=head;
for(int i=1;i<n-k;i++)
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,4,5]


leetcode.com/problems/sort-list/submissions/1542123176/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
All Submissions
Accepted 30 / 30 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:13
Editorial
Solution
Runtime
47 ms | Beats 40.95%
Analyze Complexity
Memory
75.58 MB | Beats 46.42%
Code
C++
Auto
class Solution {
public:
ListNode*Mid(ListNode*head)
{
ListNode*slow=head;
ListNode*fast=head;
while(fast->next!=NULL && fast->next->next!=NULL)
{
slow=slow->next;
fast=fast->next->next;
}
return slow;
}
ListNode*MergeSortedList(ListNode*p1,ListNode*p2)
{
if(p1==NULL && p2==NULL)
{
return (p2==NULL)? p1:p2;
}
ListNode*ans=new ListNode(0);
ListNode*ptr=ans;
while(p1!=NULL && p2!=NULL)
{
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[4,2,1,3]
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;


leetcode.com/problems/linked-list-cycle-ii/submissions/1542123990/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 18 / 18 testcases passed
Ashish Kumar submitted at Feb 14, 2025 00:13
Editorial
Solution
Runtime
3 ms | Beats 98.27%
Analyze Complexity
Memory
11.39 MB | Beats 53.99%
Code
C++
Auto
class Solution {
public:
ListNode *detectCycle(ListNode *head) {
if(head==NULL)
{
return head;
}
ListNode*fast=head;
ListNode*slow=head;
while(fast!=NULL && fast->next!=NULL)
{
fast=fast->next->next;
slow=slow->next;
if(fast==slow)
{
break;
}
}
if(fast==NULL || fast->next==NULL)
{
return NULL;
}
slow=head;
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[3,2,0,-4]
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;