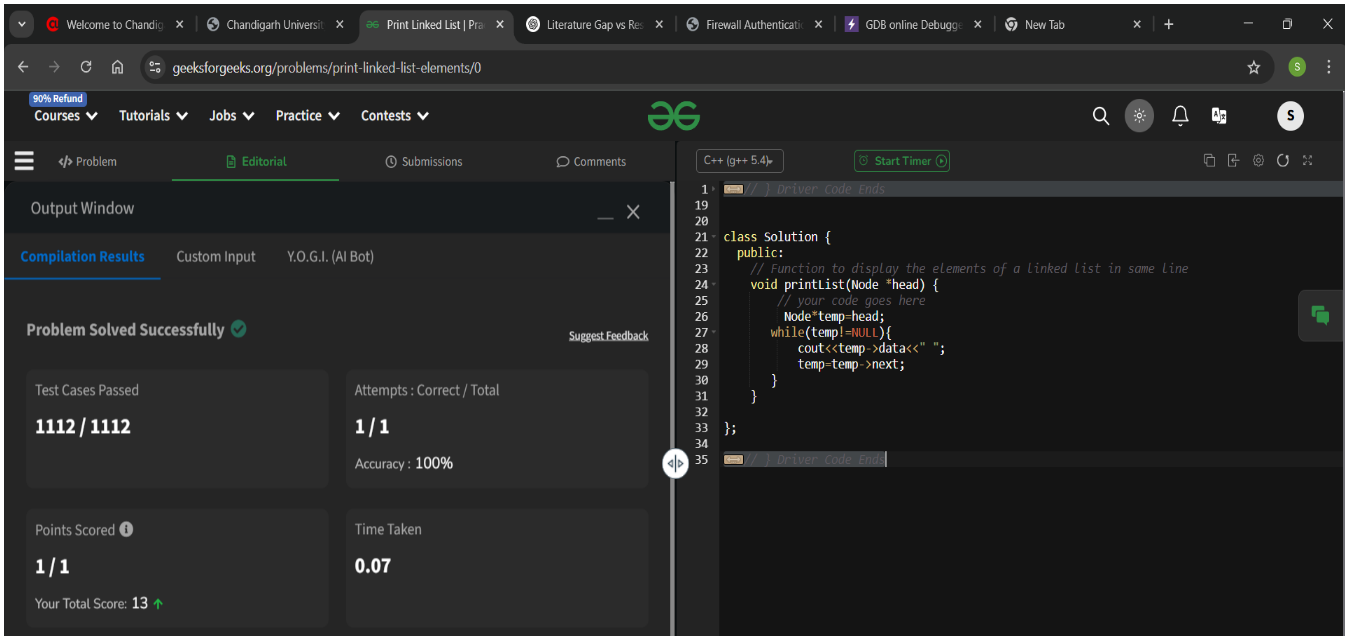
Assignment 2

Name: Sargam Anand	**UID:** 22BCS14851
Branch: BE-CSE	**Section/Group:** 607/B
Semester: 6th	**Date of Performance:** 05/03/25
Subject Name: Advanced Programming	**Subject Code:** 22CSP-351

Code 1

```
class Solution {
 public:
   void printList(Node *head) {
       Node*temp=head;
       while(temp!=NULL){
           cout<<temp->data<<" ";
           temp=temp->next;
       }
   }
};
```



geeksforgeeks.org/problems/print-linked-list-elements/0

Problem Solved Successfully

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
1 / 1
Accuracy : 100%

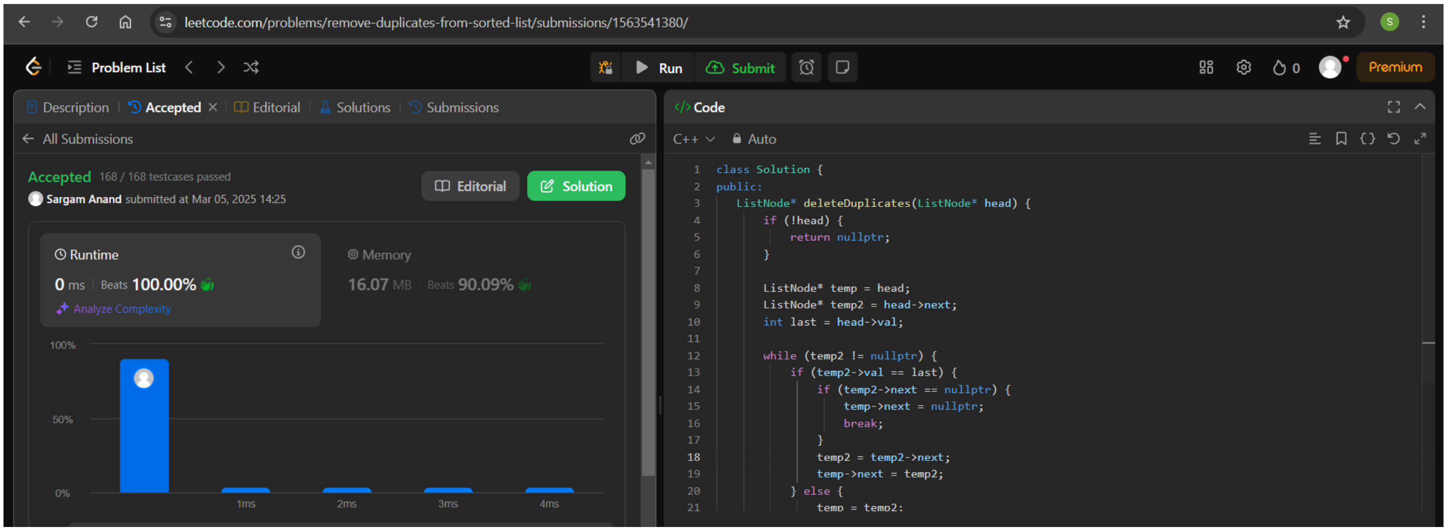
Points Scored
1 / 1
Your Total Score: 13

Time Taken
0.07

```
// } Driver Code Ends


class Solution {
  public:
    // Function to display the elements of a linked list in same line
    void printList(Node *head) {
        // your code goes here
         Node*temp=head;
        while(temp!=NULL){
            cout<<temp->data<<" ";
            temp=temp->next;
        }
    }

};

// } Driver Code Ends
```

Code 2

```
class Solution {
public:
  ListNode* deleteDuplicates(ListNode* head) {
      if (!head) {
          return nullptr;
      }

      ListNode* temp = head;
      ListNode* temp2 = head->next;
      int last = head->val;

      while (temp2 != nullptr) {
          if (temp2->val == last) {
              if (temp2->next == nullptr) {
                  temp->next = nullptr;
                  break;
              }
              temp2 = temp2->next; /
              temp->next = temp2;
          } else {
              temp = temp2;
              last = temp->val;
              temp2 = temp2->next;
          }
      }
      return head;
  }
};
```



leetcode.com/problems/remove-duplicates-from-sorted-list/submissions/1563541380/

Accepted 168 / 168 testcases passed

Sargam Anand submitted at Mar 05, 2025 14:25

Runtime

0 ms | Beats 100.00%

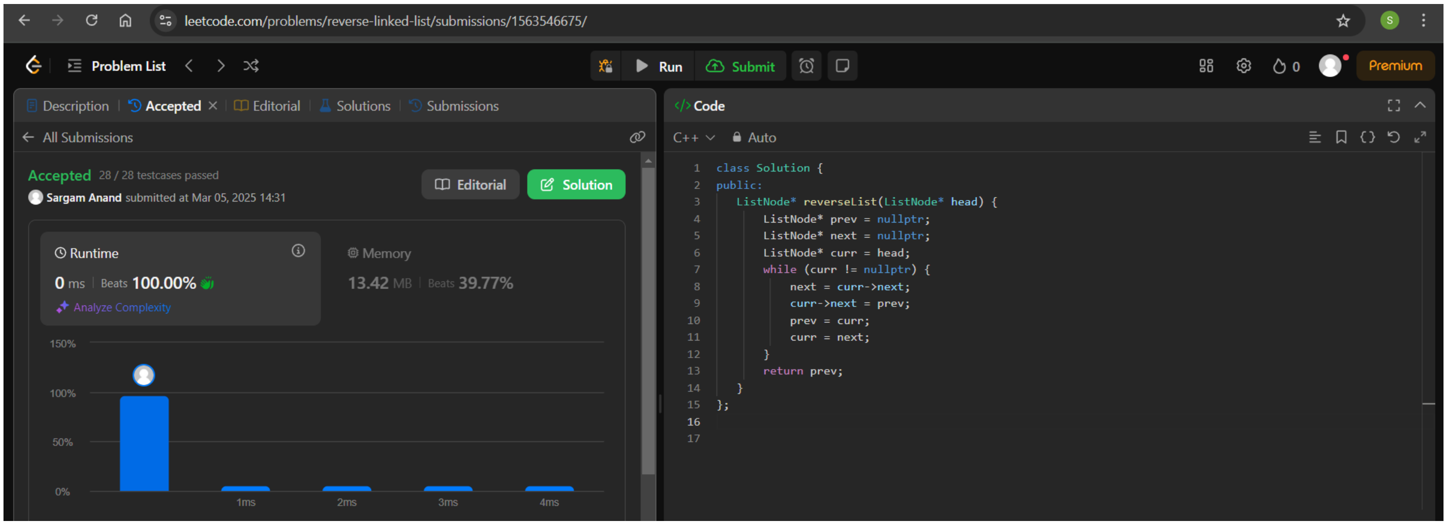
Memory

16.07 MB | Beats 90.09%

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (!head) {
            return nullptr;
        }

        ListNode* temp = head;
        ListNode* temp2 = head->next;
        int last = head->val;

        while (temp2 != nullptr) {
            if (temp2->val == last) {
                if (temp2->next == nullptr) {
                    temp->next = nullptr;
                    break;
                }
                temp2 = temp2->next;
                temp->next = temp2;
            } else {
                temp = temp2:
```

Code 3

```
class Solution {
public:
  ListNode* reverseList(ListNode* head) {
      ListNode* prev = nullptr;
      ListNode* next = nullptr;
      ListNode* curr = head;
      while (curr != nullptr) {
          next = curr->next;
          curr->next = prev;
          prev = curr;
          curr = next;
      }
      return prev;
  }
};
```



leetcode.com/problems/reverse-linked-list/submissions/1563546675/

Accepted 28 / 28 testcases passed

Sargam Anand submitted at Mar 05, 2025 14:31

Runtime 0 ms Beats 100.00%

Memory 13.42 MB Beats 39.77%

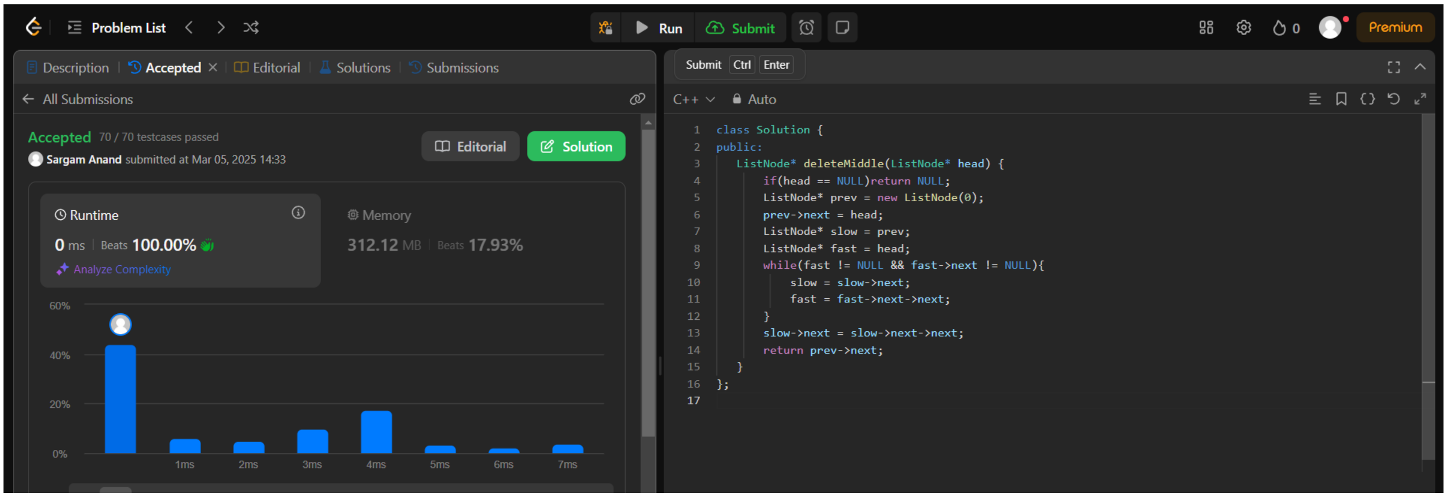
C++ Auto

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* next = nullptr;
        ListNode* curr = head;
        while (curr != nullptr) {
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
};
```

Code 4

```
class Solution {
public:
  ListNode* deleteMiddle(ListNode* head) {
      if(head == NULL)return NULL;
      ListNode* prev = new ListNode(0);
      prev->next = head;
```

```
 ListNode* slow = prev;
    ListNode* fast = head;
    while(fast != NULL && fast->next != NULL){
       slow = slow->next;
       fast = fast->next->next;
    }
    slow->next = slow->next->next;
    return prev->next;
  }
};
```



Accepted 70 / 70 testcases passed
Sargam Anand submitted at Mar 05, 2025 14:33

0 ms | Beats 100.00%

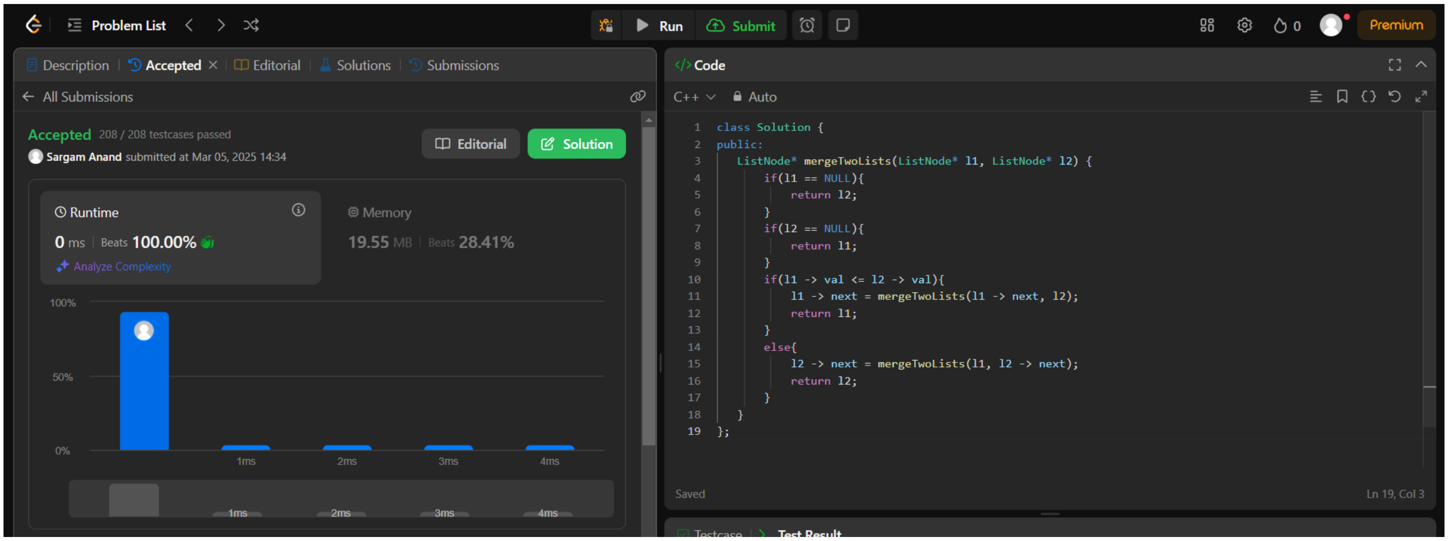
312.12 MB | Beats 17.93%

```
class Solution {
public:
   ListNode* deleteMiddle(ListNode* head) {
       if(head == NULL)return NULL;
       ListNode* prev = new ListNode(0);
       prev->next = head;
       ListNode* slow = prev;
       ListNode* fast = head;
       while(fast != NULL && fast->next != NULL){
           slow = slow->next;
           fast = fast->next->next;
       }
       slow->next = slow->next->next;
       return prev->next;
   }
};
```

Code 5

```
class Solution {
public:
  ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
     if(l1 == NULL){
        return l2;
     }
     if(l2 == NULL){
        return l1;
     }
     if(l1 -> val <= l2 -> val){
        l1 -> next = mergeTwoLists(l1 -> next, l2);
        return l1;
     }
```

```
            else{
                l2 -> next = mergeTwoLists(l1, l2 -> next);
                return l2;
            }
        }
    };
```



Accepted 208 / 208 testcases passed
Sargam Anand submitted at Mar 05, 2025 14:34

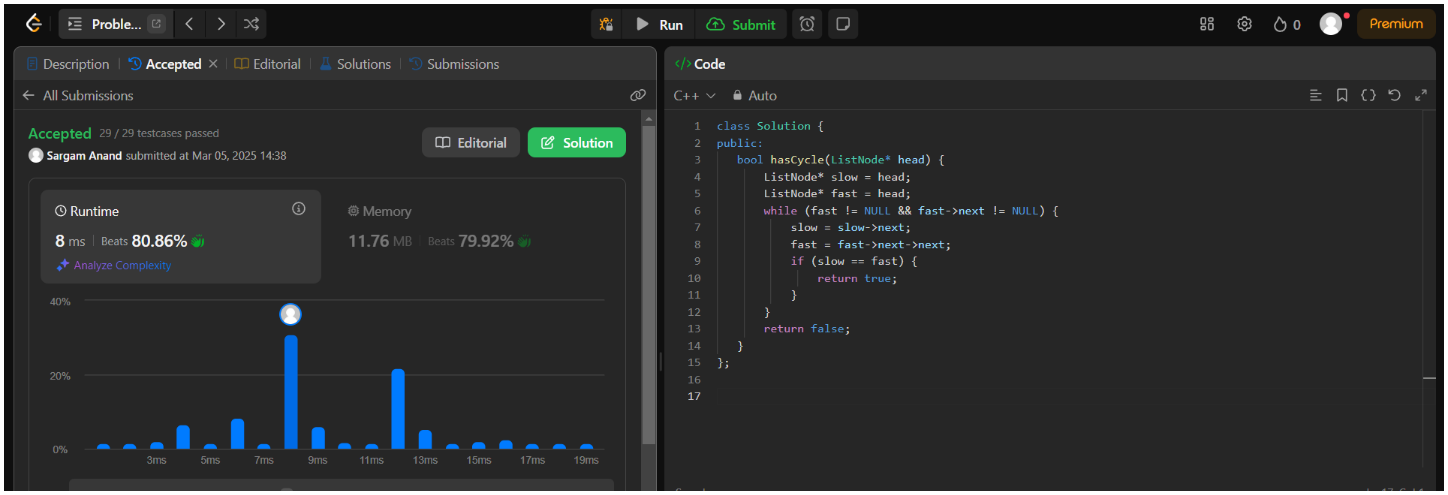
Runtime 0 ms | Beats 100.00%

Memory 19.55 MB | Beats 28.41%

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        if(l1 == NULL){
            return l2;
        }
        if(l2 == NULL){
            return l1;
        }
        if(l1 -> val <= l2 -> val){
            l1 -> next = mergeTwoLists(l1 -> next, l2);
            return l1;
        }
        else{
            l2 -> next = mergeTwoLists(l1, l2 -> next);
            return l2;
        }
    }
};
```

Code 6

```
class Solution {
public:
   bool hasCycle(ListNode* head) {
       ListNode* slow = head;
       ListNode* fast = head;
       while (fast != NULL && fast->next != NULL) {
           slow = slow->next;
           fast = fast->next->next;
           if (slow == fast) {
               return true;
           }
       }
       return false;
   }
};
```



Accepted 29 / 29 testcases passed
Sargam Anand submitted at Mar 05, 2025 14:38

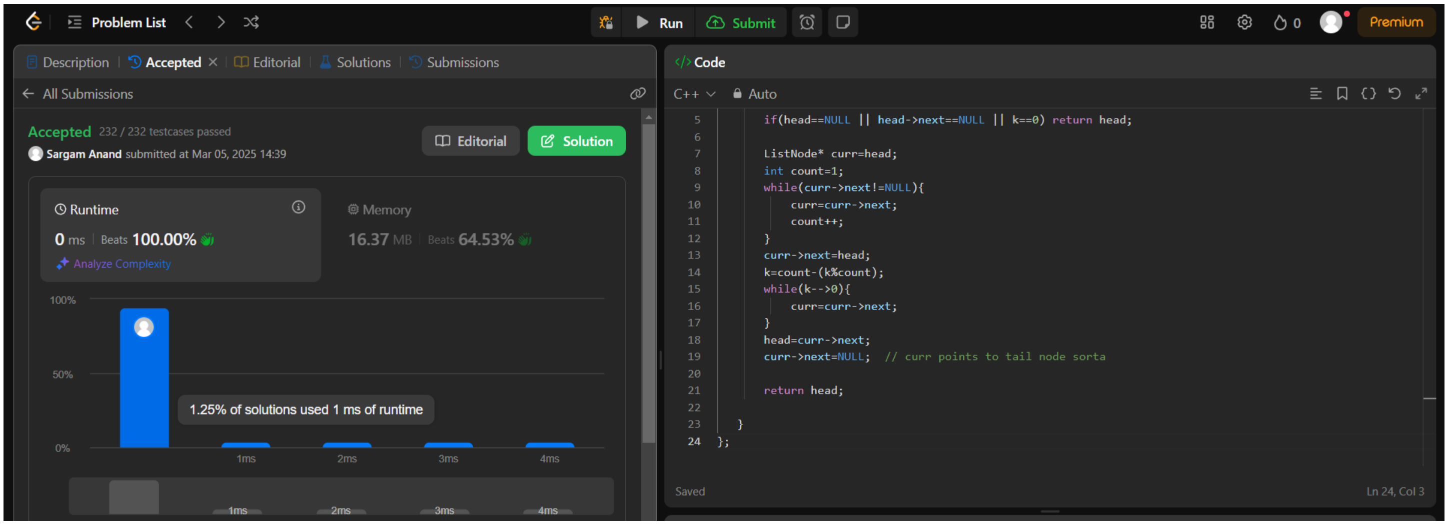
Runtime
8 ms | Beats 80.86%

Memory
11.76 MB | Beats 79.92%

```
class Solution {
public:
    bool hasCycle(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;
        while (fast != NULL && fast->next != NULL) {
            slow = slow->next;
            fast = fast->next->next;
            if (slow == fast) {
                return true;
            }
        }
        return false;
    }
};
```

Code 7

```
class Solution {
public:
   ListNode* rotateRight(ListNode* head, int k) {
       // base condition
       if(head==NULL || head->next==NULL || k==0) return head;

       ListNode* curr=head;
       int count=1;
       while(curr->next!=NULL){
           curr=curr->next;
           count++;
       }
       curr->next=head;
       k=count-(k%count);
       while(k-->0){
           curr=curr->next;
       }
       head=curr->next;
       curr->next=NULL;  // curr points to tail node sorta

       return head;

   }
};
```



Accepted 232 / 232 testcases passed

Sargam Anand submitted at Mar 05, 2025 14:39

Runtime

0 ms | Beats 100.00%

Memory

16.37 MB | Beats 64.53%

1.25% of solutions used 1 ms of runtime

```
        if(head==NULL || head->next==NULL || k==0) return head;

        ListNode* curr=head;
        int count=1;
        while(curr->next!=NULL){
            curr=curr->next;
            count++;
        }
        curr->next=head;
        k=count-(k%count);
        while(k-->0){
            curr=curr->next;
        }
        head=curr->next;
        curr->next=NULL;  // curr points to tail node sorta

        return head;

    }
};
```

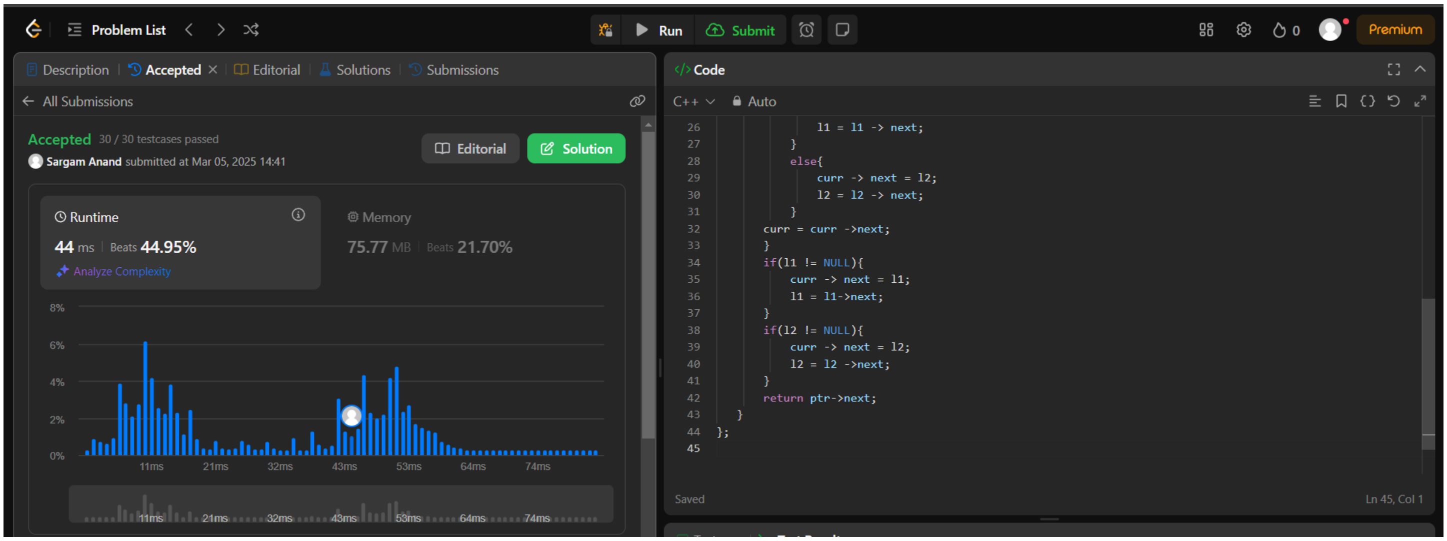
Saved

Ln 24, Col 3

Code 8

```
class Solution {
public:
   ListNode* sortList(ListNode* head) {
       if(head == NULL || head ->next == NULL)
           return head;
       ListNode *temp = NULL;
       ListNode *slow = head;
       ListNode *fast = head;
       while(fast !=  NULL && fast -> next != NULL){
           temp = slow;
           slow = slow->next;
           fast = fast ->next ->next;

       }
       temp -> next = NULL;
       ListNode* l1 = sortList(head);
       ListNode* l2 = sortList(slow);
       return mergelist(l1, l2);
   }
```

```
    ListNode* mergelist(ListNode *l1, ListNode *l2){
        ListNode *ptr = new ListNode(0);
        ListNode *curr = ptr;
        while(l1 != NULL && l2 != NULL){
            if(l1->val <= l2->val){
                curr -> next = l1;
                l1 = l1 -> next;
            }
            else{
                curr -> next = l2;
                l2 = l2 -> next;
            }
        curr = curr ->next;
        }
        if(l1 != NULL){
            curr -> next = l1;
            l1 = l1->next;
        }
        if(l2 != NULL){
            curr -> next = l2;
            l2 = l2 ->next;
        }
        return ptr->next;
    }
};
```



Accepted 30 / 30 testcases passed

Sargam Anand submitted at Mar 05, 2025 14:41

Runtime

44 ms | Beats 44.95%

Analyze Complexity

Memory

75.77 MB | Beats 21.70%

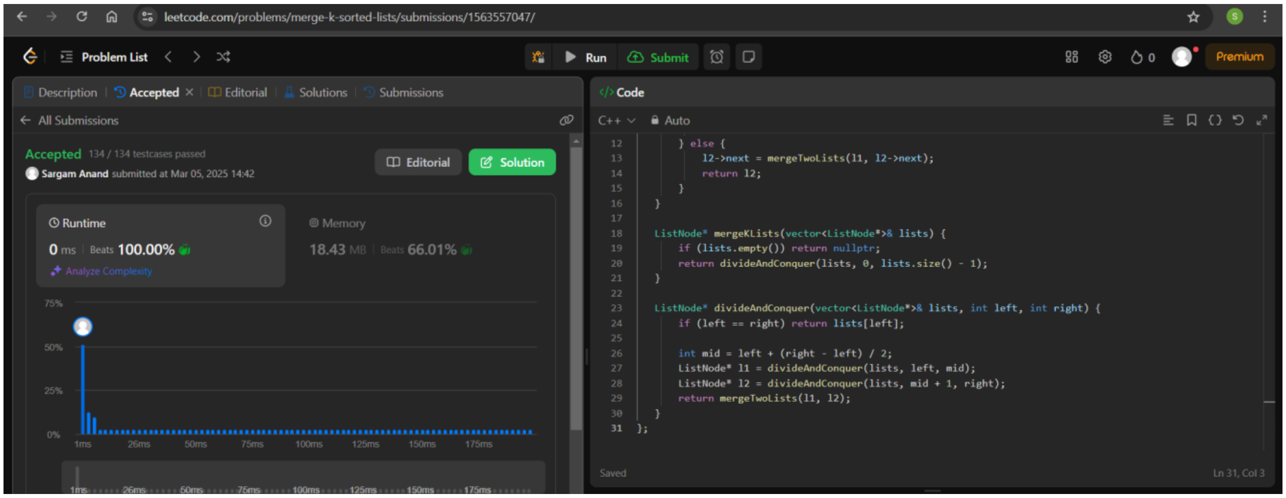
```
                l1 = l1 -> next;
            }
            else{
                curr -> next = l2;
                l2 = l2 -> next;
            }
        curr = curr ->next;
        }
        if(l1 != NULL){
            curr -> next = l1;
            l1 = l1->next;
        }
        if(l2 != NULL){
            curr -> next = l2;
            l2 = l2 ->next;
        }
        return ptr->next;
    }
};
```

Saved

Ln 45, Col 1

Code 9

```
#include <vector>
using namespace std;
class Solution {
public:
   ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
       if (!l1) return l2;
       if (!l2) return l1;

       if (l1->val < l2->val) {
           l1->next = mergeTwoLists(l1->next, l2);
           return l1;
       } else {
           l2->next = mergeTwoLists(l1, l2->next);
           return l2;
       }
   }

   ListNode* mergeKLists(vector<ListNode*>& lists) {
       if (lists.empty()) return nullptr;
       return divideAndConquer(lists, 0, lists.size() - 1);
   }

   ListNode* divideAndConquer(vector<ListNode*>& lists, int left, int right) {
       if (left == right) return lists[left];

       int mid = left + (right - left) / 2;
       ListNode* l1 = divideAndConquer(lists, left, mid);
       ListNode* l2 = divideAndConquer(lists, mid + 1, right);
       return mergeTwoLists(l1, l2);
   }
};
```



Accepted 134 / 134 testcases passed

Sargam Anand submitted at Mar 05, 2025 14:42

Runtime

0 ms | Beats 100.00%

Memory

18.43 MB | Beats 66.01%

```
        } else {
            l2->next = mergeTwoLists(l1, l2->next);
            return l2;
        }
    }

    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) return nullptr;
        return divideAndConquer(lists, 0, lists.size() - 1);
    }

    ListNode* divideAndConquer(vector<ListNode*>& lists, int left, int right) {
        if (left == right) return lists[left];

        int mid = left + (right - left) / 2;
        ListNode* l1 = divideAndConquer(lists, left, mid);
        ListNode* l2 = divideAndConquer(lists, mid + 1, right);
        return mergeTwoLists(l1, l2);
    }
};
```