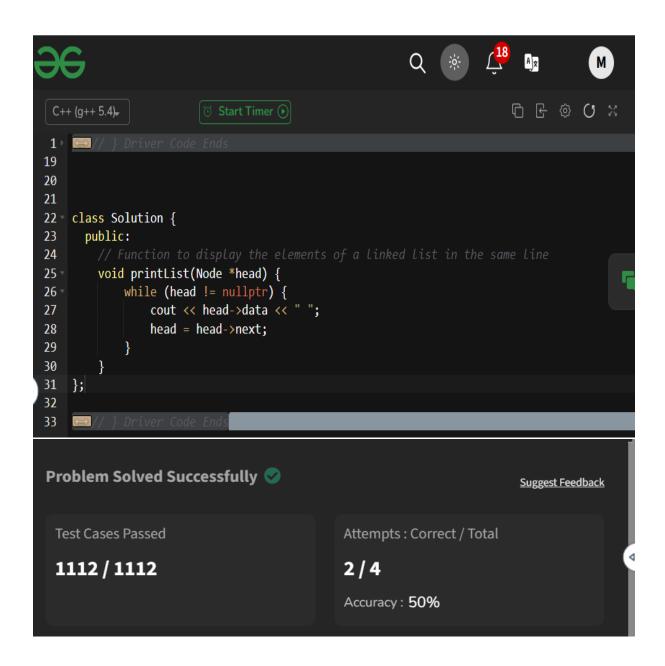
AP ASSIGNMENT-2

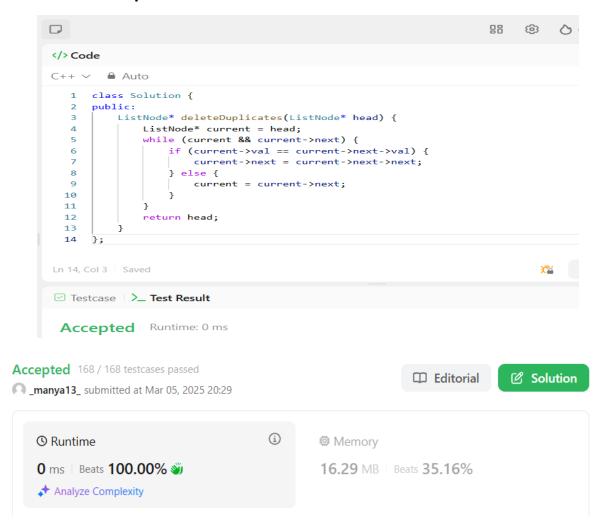
Name- Manya Goyal

UID- 22BCS16483

1. Print Linked List:

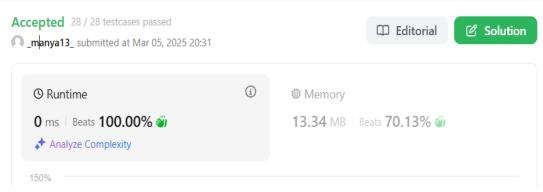


2. Remove duplicates from a sorted list:

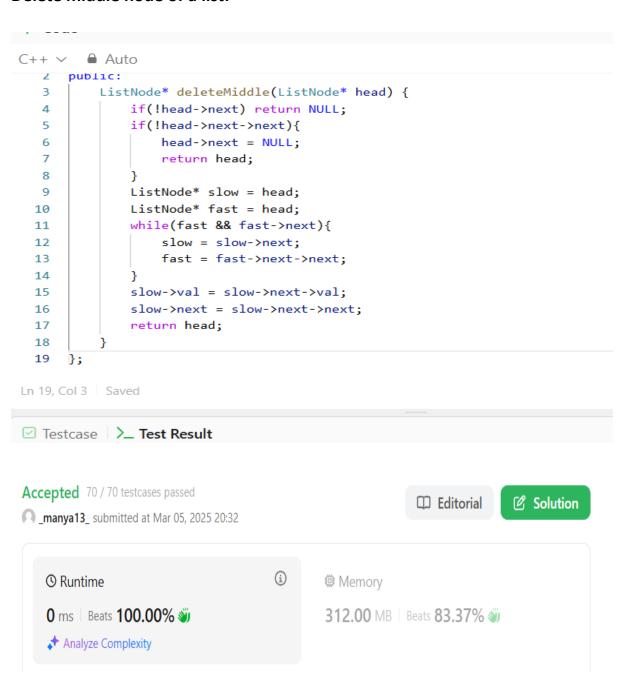


3. Reverse a linked list:

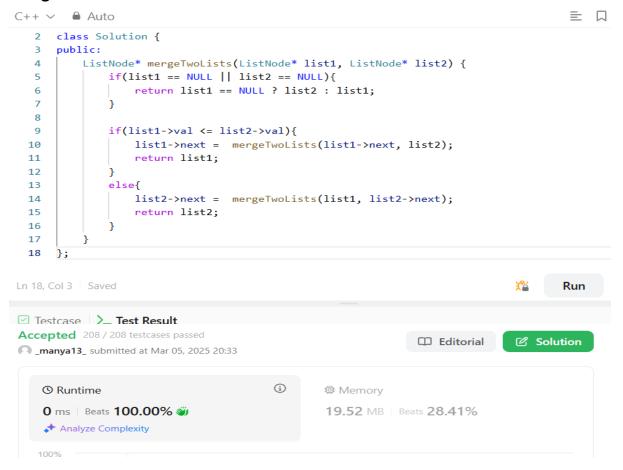
```
y couc
C++ ∨ Auto
                                                                                     = □ (
  2 class Solution {
      public:
  4
          ListNode* reverseList(ListNode* head) {
  5
              ListNode* prev = NULL;
  6
  7
              ListNode* curr = head;
  8
  9
              while(curr != NULL){
 10
                 ListNode* forward = curr->next;
                  curr->next = prev;
 11
 12
                  prev = curr;
 13
                  curr = forward;
 14
 15
 16
              return prev;
 17
Ln 18, Col 3 | Saved
                                                                             *
                                                                                    Run
✓ Testcase  \>_ Test Result
```



4. Delete middle node of a list:



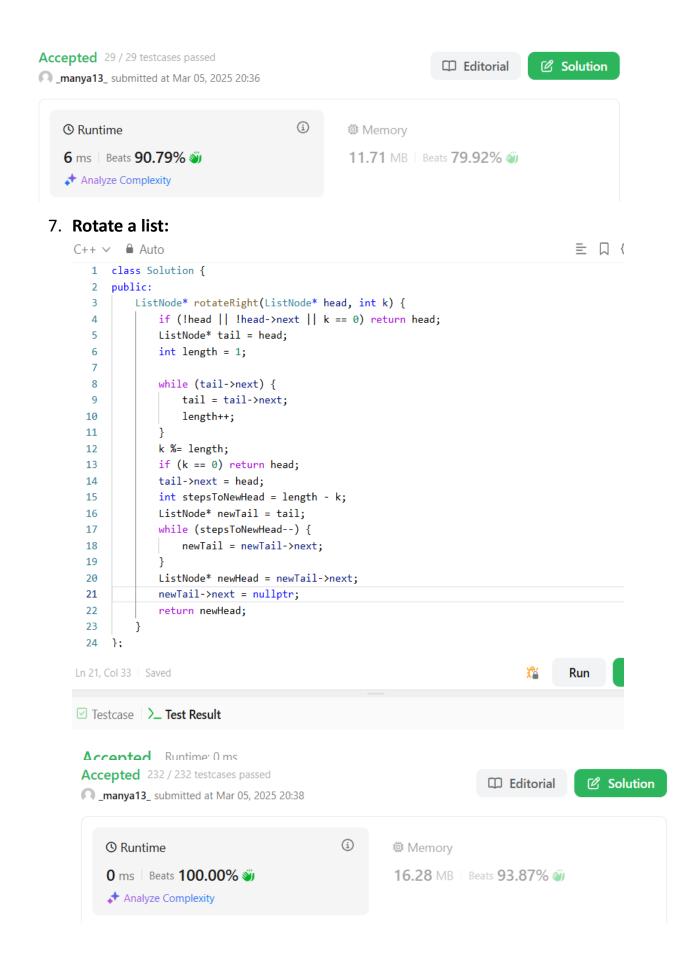
5. Merge two sorted linked lists:



6. Detect a cycle in a linked list:

```
C++ ∨ Auto
                                                                                      \equiv
     */
   8
   9
      class Solution {
  10
      public:
          bool hasCycle(ListNode* head) {
  11
              ListNode* slow = head;
 12
              ListNode* fast = head;
 13
 14
              while (fast != NULL && fast->next != NULL) {
 15
                  slow = slow->next;
 16
                  fast = fast->next->next;
 17
                  if (slow == fast) {
  18
                      return true;
  19
  20
  21
              return false;
  22
      };
  23
                                                                                     Run
Ln 23, Col 3 | Saved
✓ Testcase  \>_ Test Result
```

Accepted Runtime: 0 ms



8. Sort List:

```
public:
  ListNode* merge(ListNode* list1, ListNode* list2) {
         ListNode* C = new ListNode(100);
         ListNode* temp = C;
         while(list1!= NULL && list2!=NULL){
              if(list1->val <= list2->val){
                  temp->next = list1;
                 list1 = list1->next;
                 temp = temp ->next ;
              }
             else{
                 temp->next = list2;
                 list2 = list2->next;
                 temp = temp->next ;
         if(list1 == NULL )temp->next = list2;
         if(list2==NULL) temp->next = list1;
      return C->next ;
      ListNode* sortList(ListNode* head) {
         if(head==NULL || head->next == NULL)return head;
         // to find the middle of the linked list
         ListNode* slow = head;
         ListNode* fast = head ;
         while(fast->next!=NULL && fast->next!=NULL){
              slow = slow->next;
             fast = fast->next->next ; }
         ListNode* a = head;
         ListNode* b = slow->next;
         slow->next = NULL;
         a = sortList(a);
         b = sortList(b);
         ListNode* c = merge(a,b);
          return c ;
Accepted 30 / 30 testcases passed
                                                                            Solution
                                                            □ Editorial
nanya13_ submitted at Mar 05, 2025 20:41
                                       (i)
   © Runtime
                                               Memory
   42 ms | Beats 48.60%
                                               75.83 MB | Beats 8.72%
    ♣ Analyze Complexity
```

9. Merge k sorted lists:

```
C++ V Auto
   1 class Solution {
   2
      public:
   3
          ListNode* mergeKLists(vector<ListNode*>& lists) {
   4
              if (lists.empty()) {
   5
                  return nullptr;
   6
   7
              return mergeKListsHelper(lists, 0, lists.size() - 1);
   8
          ListNode* mergeKListsHelper(vector<ListNode*>& lists, int start, int end) {
   9
  10
              if (start == end) {
  11
                  return lists[start];
  12
              if (start + 1 == end) {
  13
                  return merge(lists[start], lists[end]);
  14
  15
  16
              int mid = start + (end - start) / 2;
  17
              ListNode* left = mergeKListsHelper(lists, start, mid);
  18
              ListNode* right = mergeKListsHelper(lists, mid + 1, end);
              return merge(left, right); }
  19
          ListNode* merge(ListNode* 11, ListNode* 12) {
  20
              ListNode* dummy = new ListNode(0);
  21
  22
              ListNode* curr = dummy;
              while (11 && 12) {
  23
  24
                  if (l1->val < l2->val) {
  25
                      curr->next = 11;
                      11 = 11->next;
  26
  27
                   } else {
  28
                      curr->next = 12;
  29
                      12 = 12->next;
  30
  31
                  curr = curr->next;
  32
  33
              curr->next = 11 ? 11 : 12;
              return dummy->next;
  35
Accepted 134 / 134 testcases passed
                                                                  ☐ Editorial
                                                                                  Solution
manya13_ submitted at Mar 05, 2025 20:43
    O Runtime
                                                  Memory
    4 ms | Beats 48.89%
                                                   23.90 MB | Beats 7.35%
    * Analyze Complexity
   75%
```