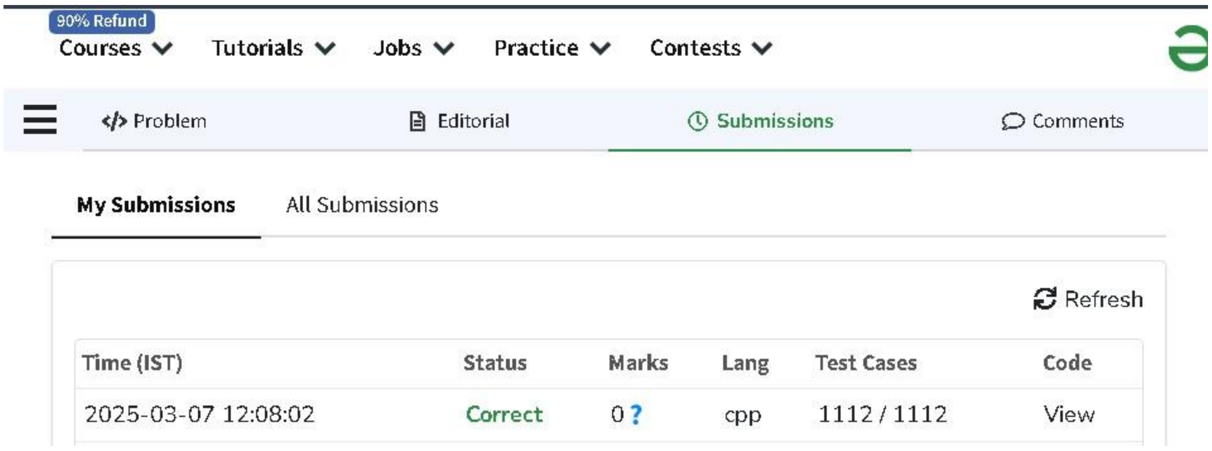
ASSIGNMENT - 3

Student Name: Priyanka Sharma | **UID: 22BCS15114**

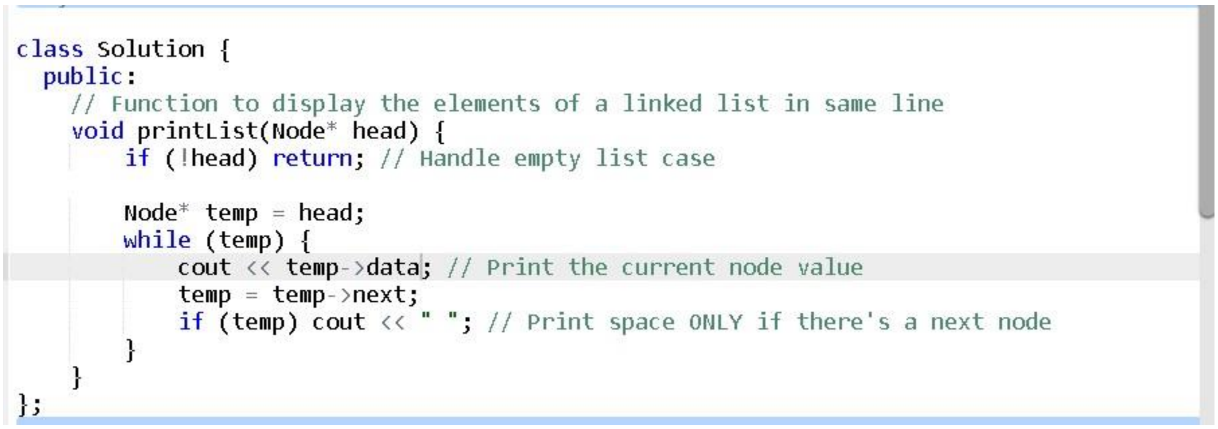
Branch: BE-CSE | **Section/Group: 608/B**

Semester: 6th | **Subject Name: AP LAB**

1.Print Linked List:

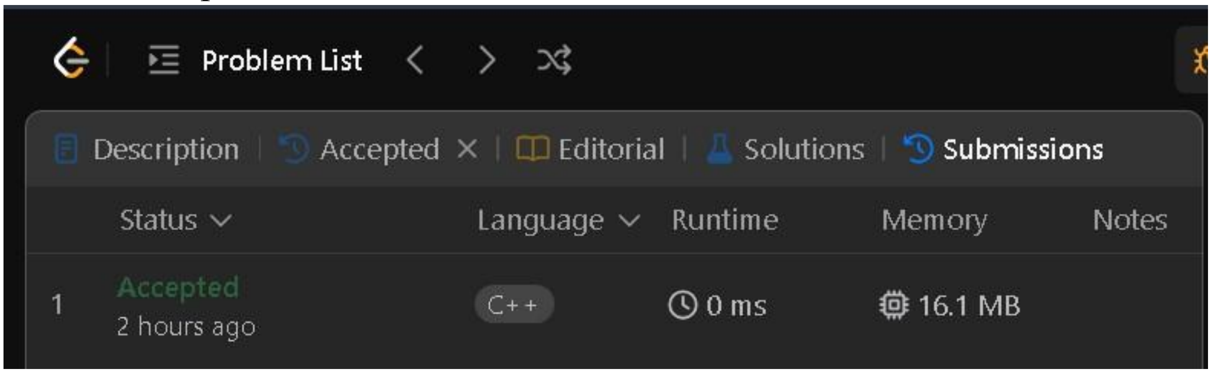


Time (IST)	Status	Marks	Lang	Test Cases	Code
2025-03-07 12:08:02	Correct	0 ?	cpp	1112 / 1112	View



```
class Solution {
  public:
    // Function to display the elements of a linked list in same line
    void printList(Node* head) {
        if (!head) return; // Handle empty list case

        Node* temp = head;
        while (temp) {
            cout << temp->data; // Print the current node value
            temp = temp->next;
            if (temp) cout << " "; // Print space ONLY if there's a next node
        }
    }
};
```

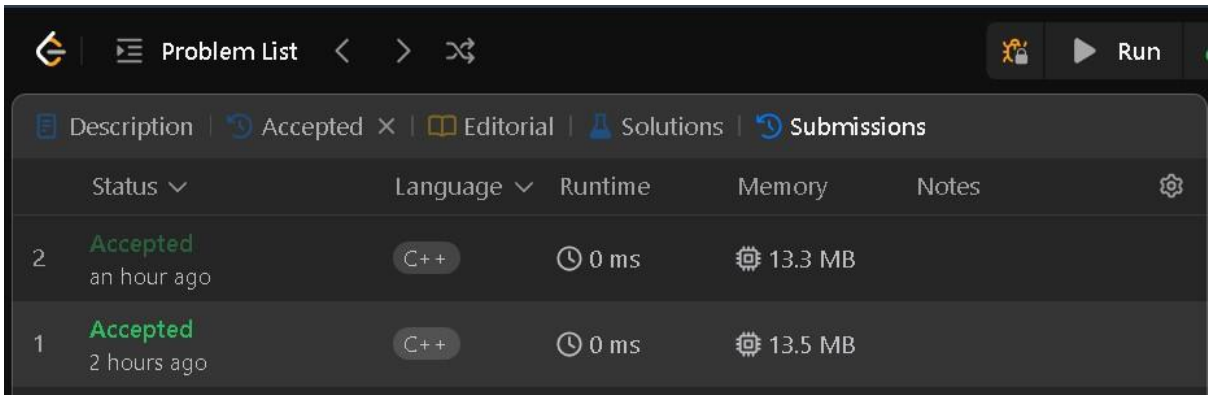
2.Remove duplicates from a sorted list:



	Status	Language	Runtime	Memory	Notes
1	Accepted 2 hours ago	C++	0 ms	16.1 MB	

```
class Solution {
public:
   ListNode* deleteDuplicates(ListNode* head) {
      ListNode* current = head;
      while(current && current-> next){
         if(current->val == current->next->val){
            current->next = current->next->next;
         } else{
            current = current->next;
         }

      }return head;
   }
};
```

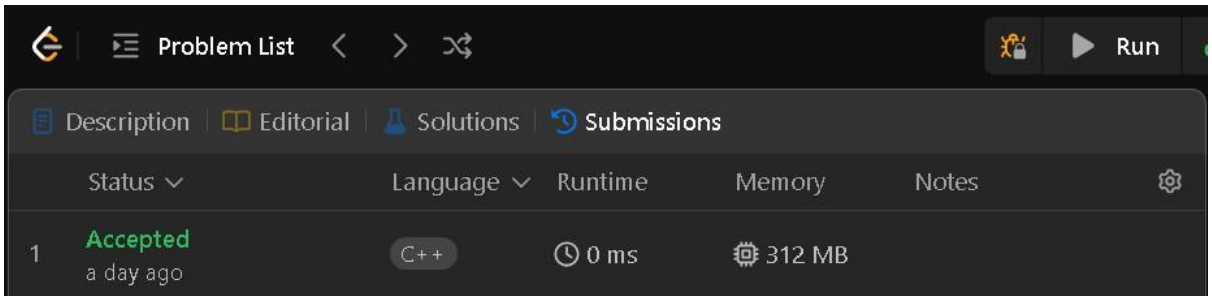
3. Reverse a linked list:



	Status	Language	Runtime	Memory	Notes
2	Accepted an hour ago	C++	0 ms	13.3 MB	
1	Accepted 2 hours ago	C++	0 ms	13.5 MB	

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

4.Delete middle node of a list:

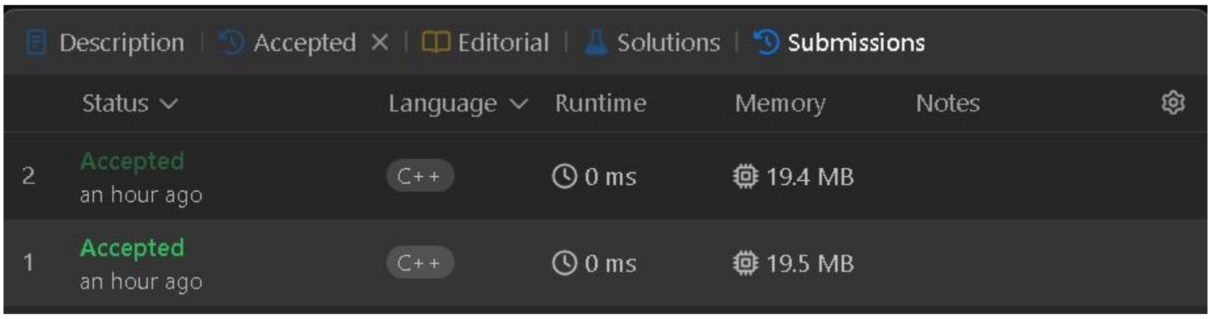


	Status	Language	Runtime	Memory	Notes
1	Accepted a day ago	C++	0 ms	312 MB	

```
class Solution {
public:
   ListNode* deleteMiddle(ListNode* head) {
      if(!head || !head->next) return nullptr;
      int count = 0;
      ListNode* temp = head;
      while(temp!=nullptr){
         count++ ;
         temp = temp->next;
      }
      count = count/2;
      temp = head;
      while(temp!=nullptr){
         count--;
         if(count==0){
            ListNode* mid = temp->next;
            temp-> next = temp->next->next;
            delete mid;
            break;
         }
         temp = temp->next;

      }
      return head;
   }
};
```

5. Merge two sorted linked lists:

```
var mergeTwoLists = function(list1, list2) {
   let mergedList = new ListNode(0, null);
   let curr = mergedList;

   // iterate through both lists and compare their current nodes
   while (list1 != null && list2 != null)
   {
      if (list1.val < list2.val)
      {
         curr.next = list1;
         list1 = list1.next;
      }
      else
      {
         curr.next = list2;
         list2 = list2.next;
      }
      curr = curr.next;
   }
   curr.next = (list1 != null) ? list1 : list2;
   return mergedList.next;
};
```



	Status	Language	Runtime	Memory	Notes
2	Accepted an hour ago	C++	0 ms	19.4 MB	
1	Accepted an hour ago	C++	0 ms	19.5 MB	

6. Detect a cycle in a linked list:

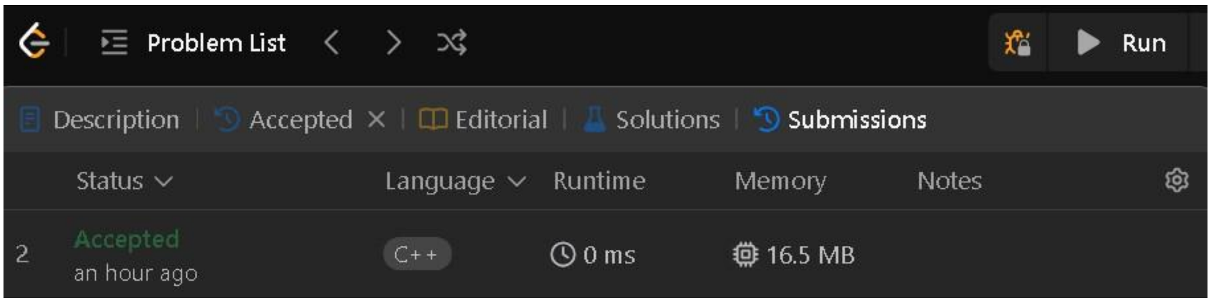
```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode* slow = head;
        ListNode* fast = head;
```

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

```
        while(fast && fast->next){
            slow = slow->next;
            fast = fast->next->next;
            if(slow == fast)return true;
        }
        return false;
    }
};
```

7.Rotate a list:

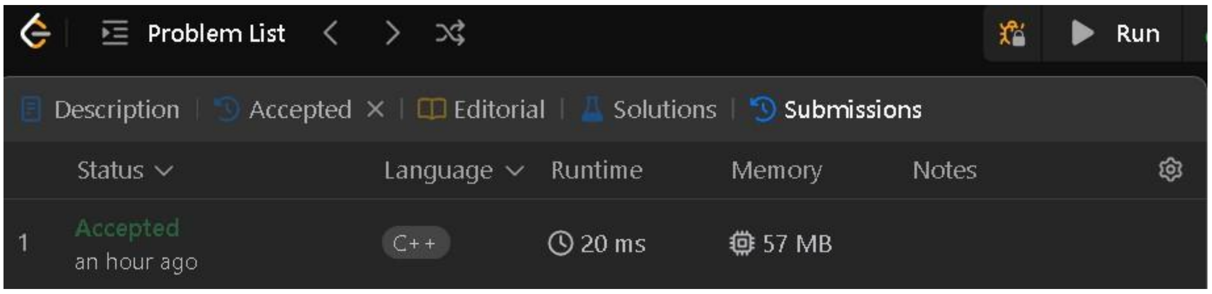
```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if(!head || !head->next || k == 0) return head;
        ListNode* temp = head;
        int length = 1;
        while(temp -> next){
            temp = temp->next;
            length++;
        }
        temp->next = head;
        k = k%length;
        if(k==0){
            temp->next = nullptr;

        }return head;
    }
};
```



	Status	Language	Runtime	Memory	Notes
2	Accepted an hour ago	C++	0 ms	16.5 MB	

8. Sort List:

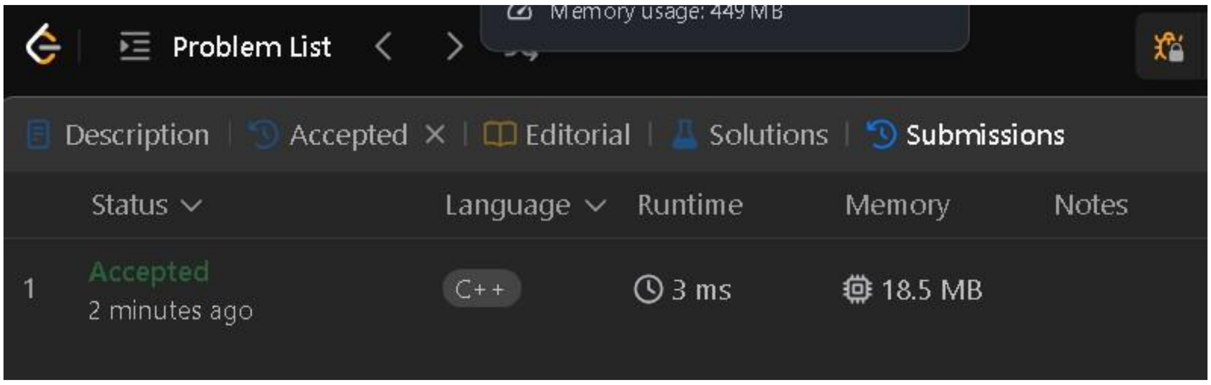
```
class Solution {
public:
   ListNode* sortList(ListNode* head) {
      class Solution {
public:
   ListNode* rotateRight(ListNode* head, int k) {
      if(!head || !head->next || k == 0) return head;
      ListNode* temp = head;
      int length = 1;
      while(temp -> next){
         temp = temp->next;
         length++;
      }
      temp->next = head;
      k = k%length;
      if(k==0){
         temp->next = nullptr;

      }return head;

   }
};
```



	Status	Language	Runtime	Memory	Notes
1	Accepted an hour ago	C++	20 ms	57 MB	

9. Merge k sorted lists:

```
class Solution {
   public ListNode mergeKLists(ListNode[] lists) {
      if (lists == null || lists.length == 0) {
         return null;
      }
      return mergeKListsHelper(lists, 0, lists.length - 1);
   }
   private ListNode mergeKListsHelper(ListNode[] lists, int start, int end) {
      if (start == end) {
         return lists[start];
      }
      if (start + 1 == end) {
         return merge(lists[start], lists[end]);
      }
      int mid = start + (end - start) / 2;
      ListNode left = mergeKListsHelper(lists, start, mid);
      ListNode right = mergeKListsHelper(lists, mid + 1, end);
      return merge(left, right);
   }
   private ListNode merge(ListNode l1, ListNode l2) {
      ListNode dummy = new ListNode(0);
      ListNode curr = dummy;
      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;
      }
      curr.next = (l1 != null) ? l1 : l2;
      return dummy.next;
   }
```



	Status	Language	Runtime	Memory	Notes
1	Accepted 2 minutes ago	C++	3 ms	18.5 MB	