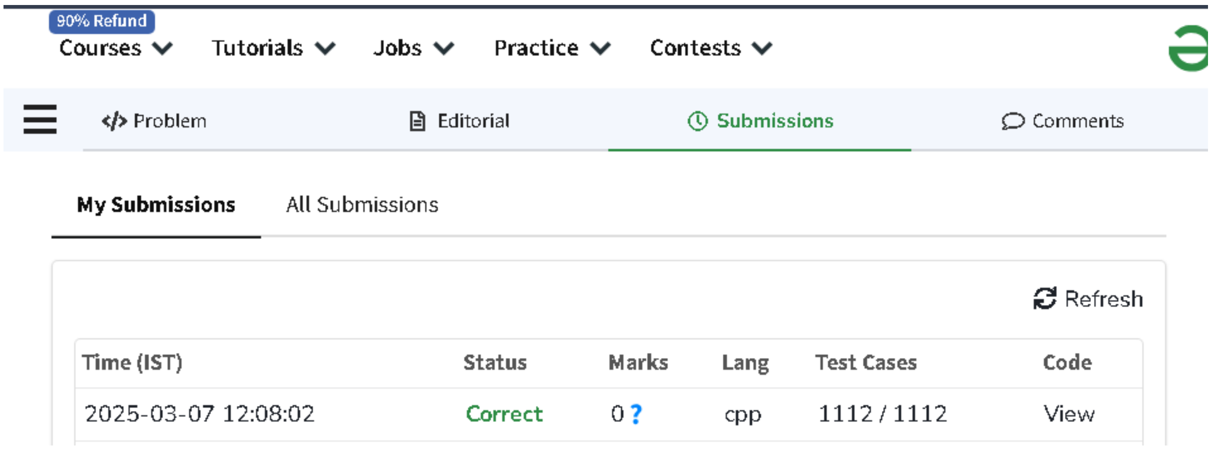
ASSIGNMENT - 3

Student Name: Manreet Kaur **UID: 22BCS15550**

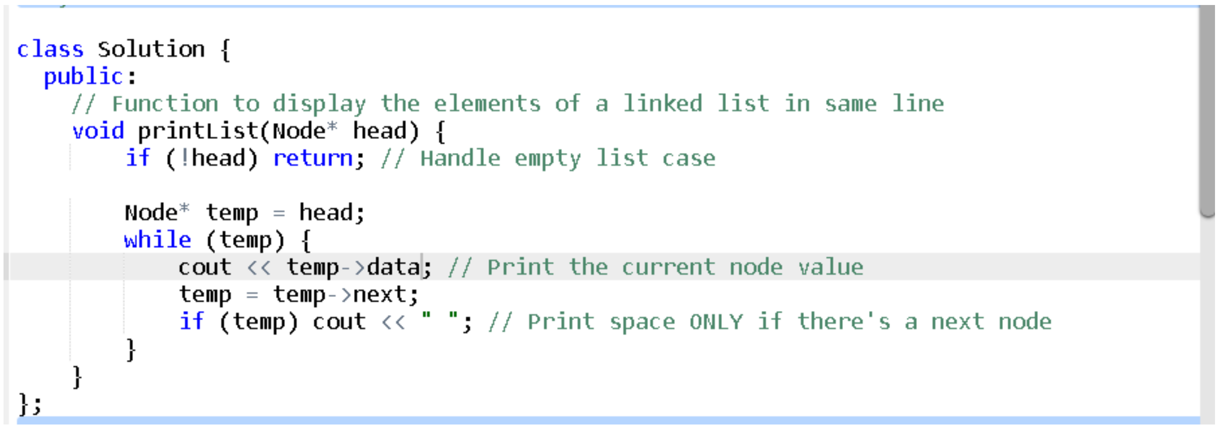
Branch: BE-CSE **Section/Group: 608/B**

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

1.Print Linked List:

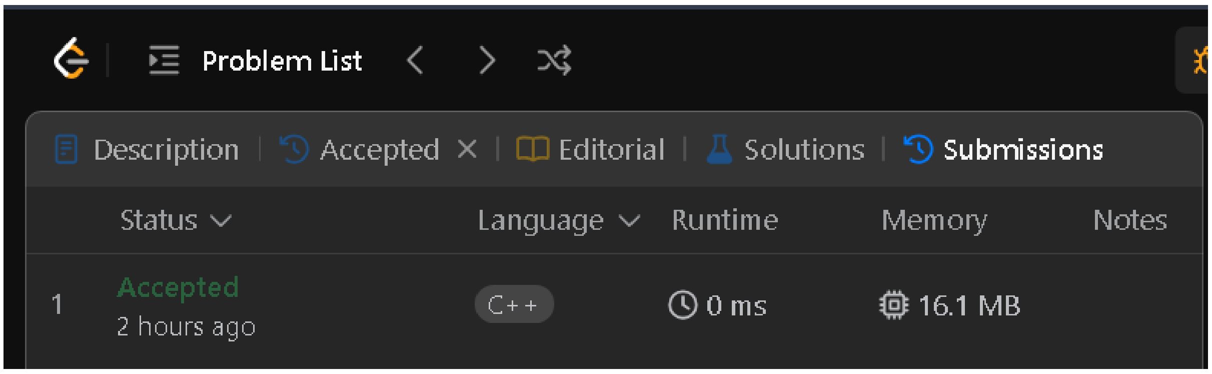


Time (IST)	Status	Marks	Lang	Test Cases	Code
2025-03-07 12:08:02	Correct	0 ?	cpp	1112 / 1112	View

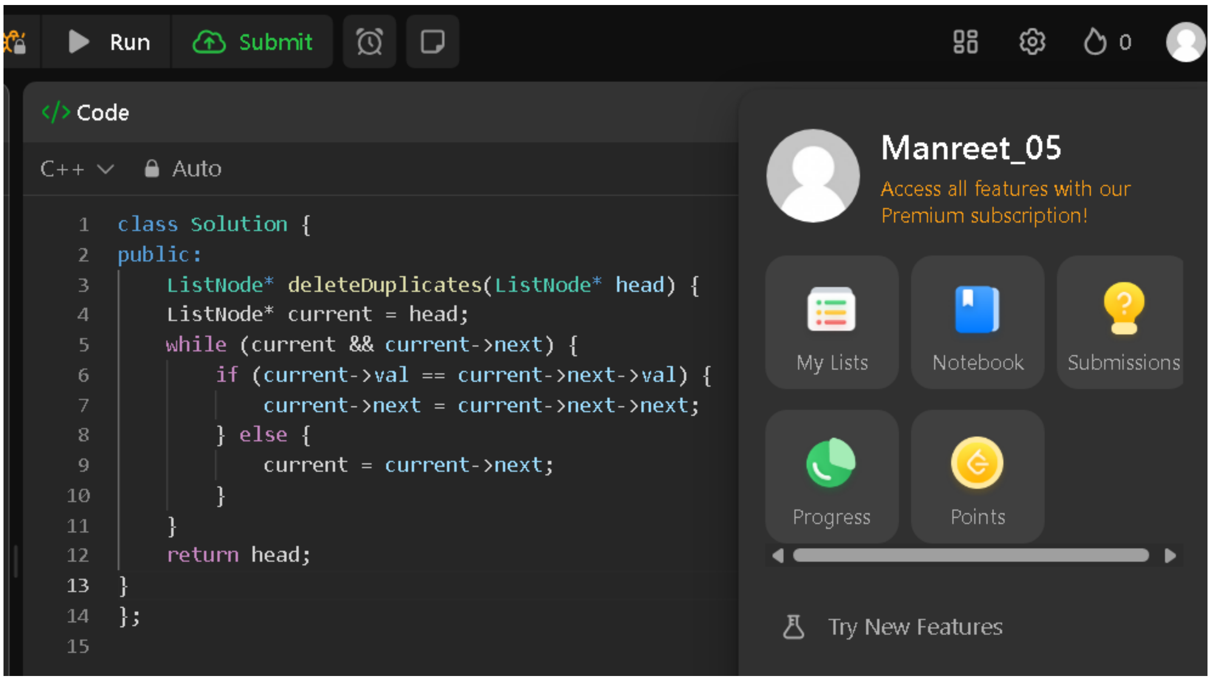


```
class Solution {
  public:
    // Function to display the elements of a linked list in same line
    void printList(Node* head) {
        if (!head) return; // Handle empty list case

        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:

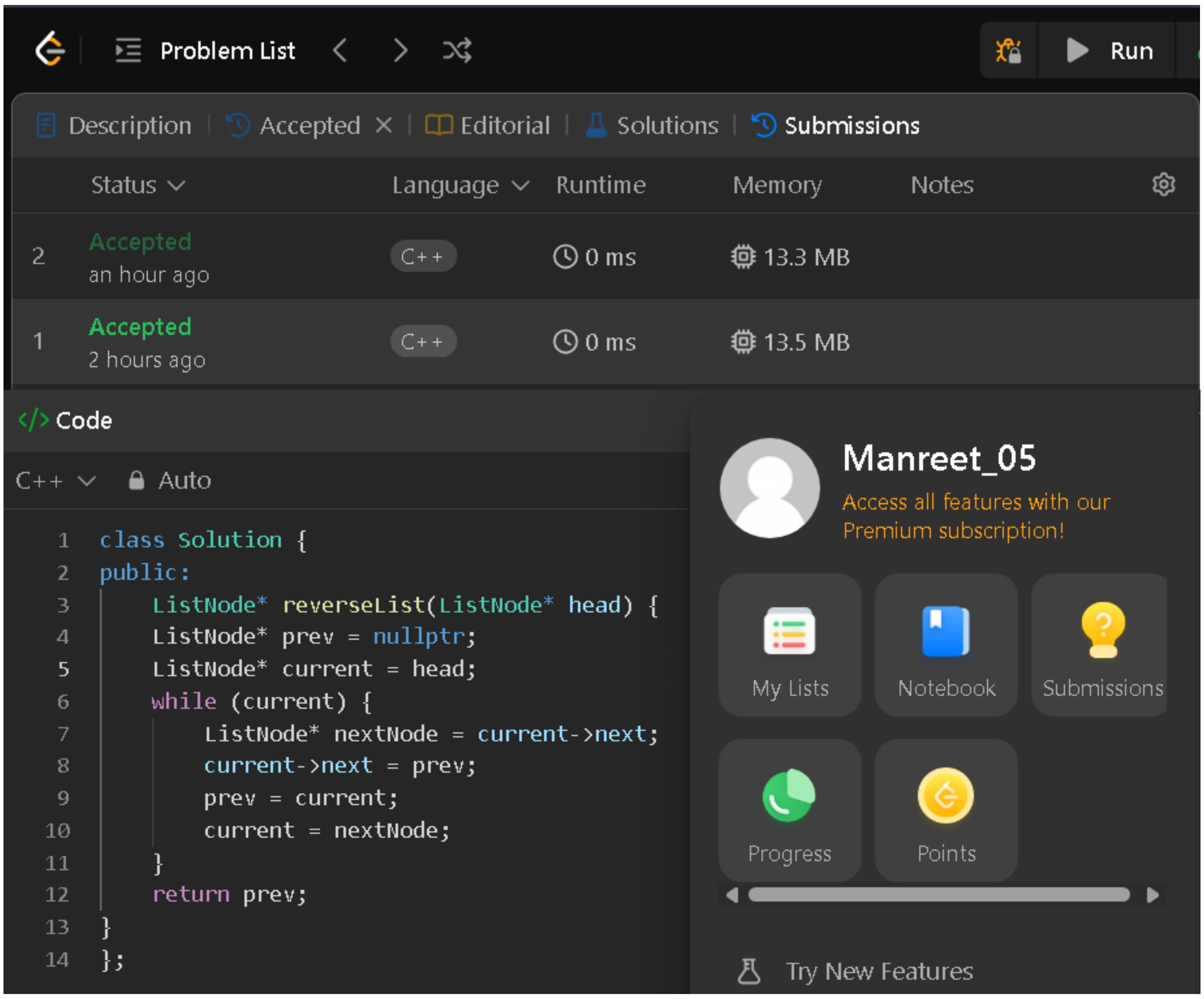


	Status	Language	Runtime	Memory	Notes
1	Accepted 2 hours ago	C++	0 ms	16.1 MB	



```cpp
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;
}
};
```

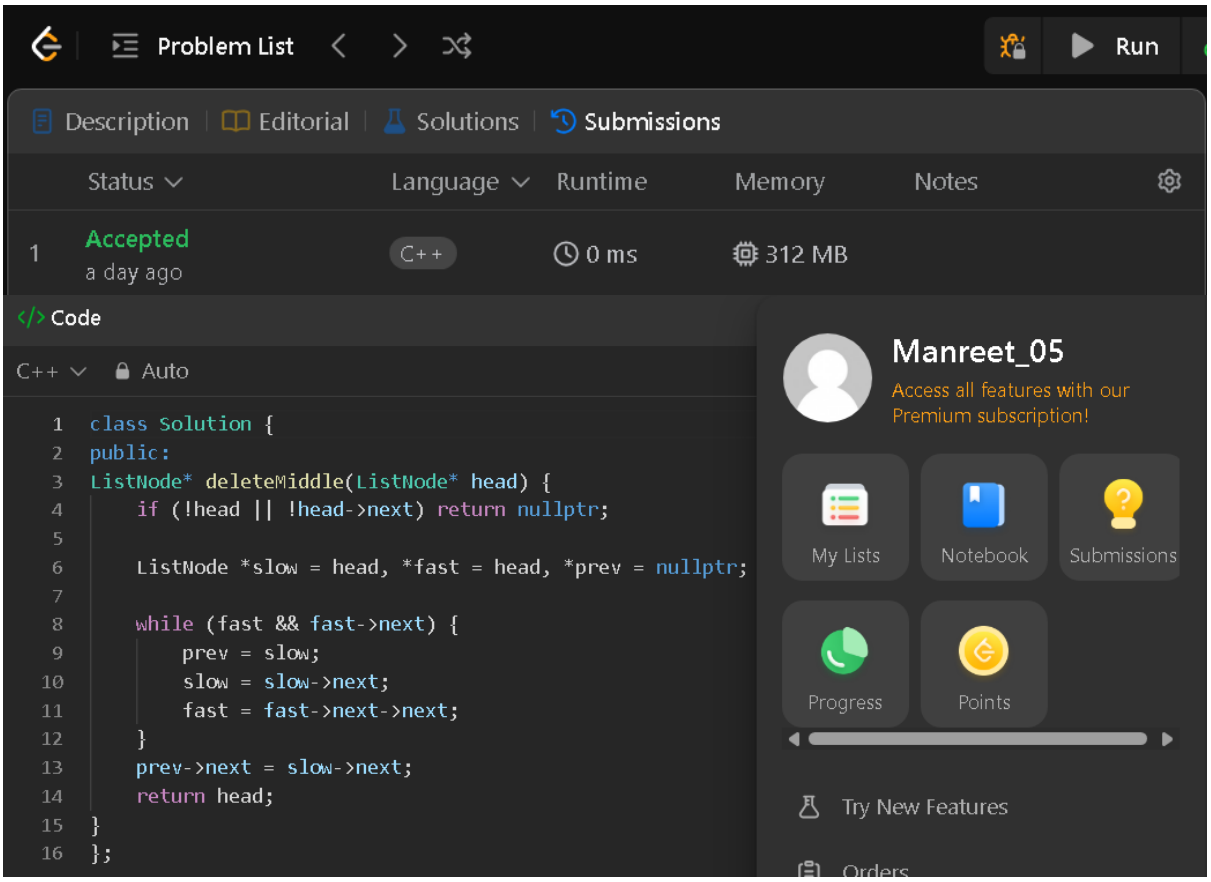
3. Reverse a linked list:



	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	

```cpp
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;
}
};
```

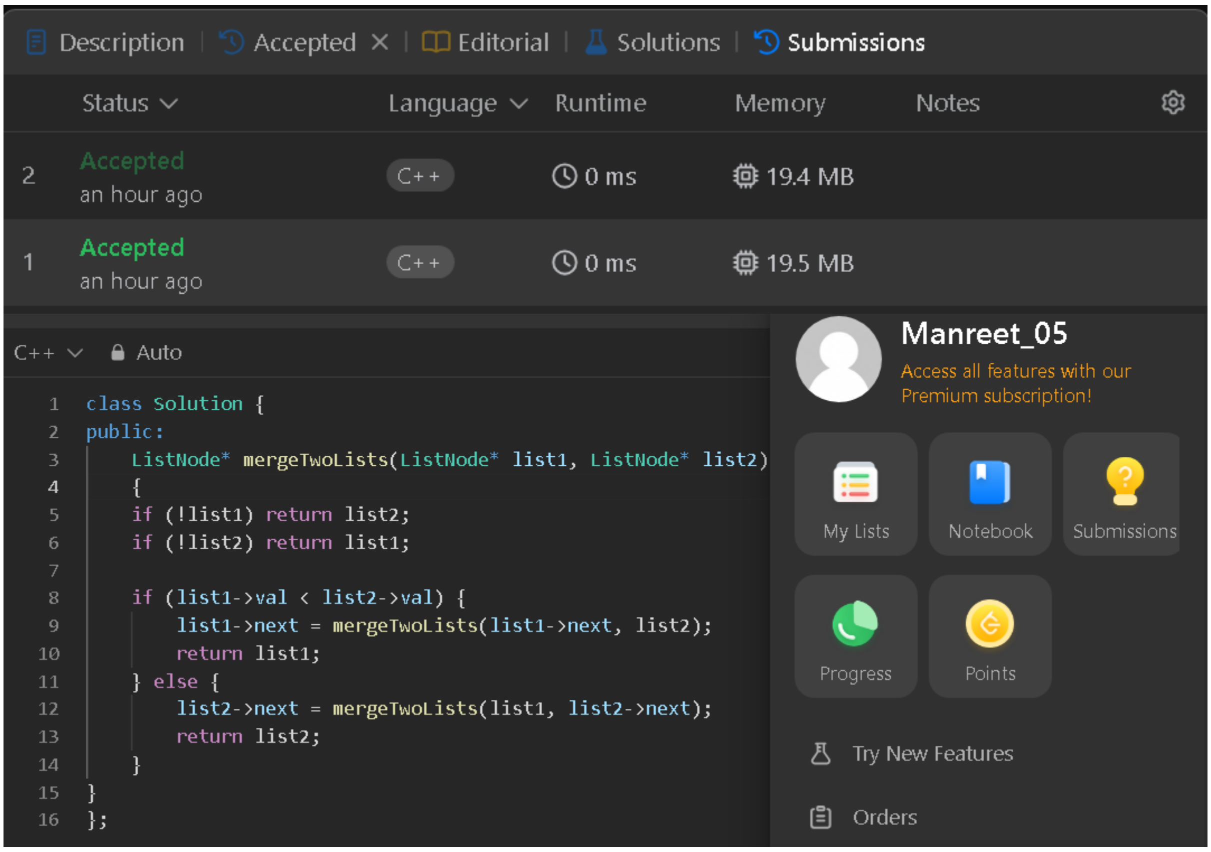
4.Delete middle node of a list:



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;

    ListNode *slow = head, *fast = head, *prev = nullptr;

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

5. Merge two sorted linked lists:



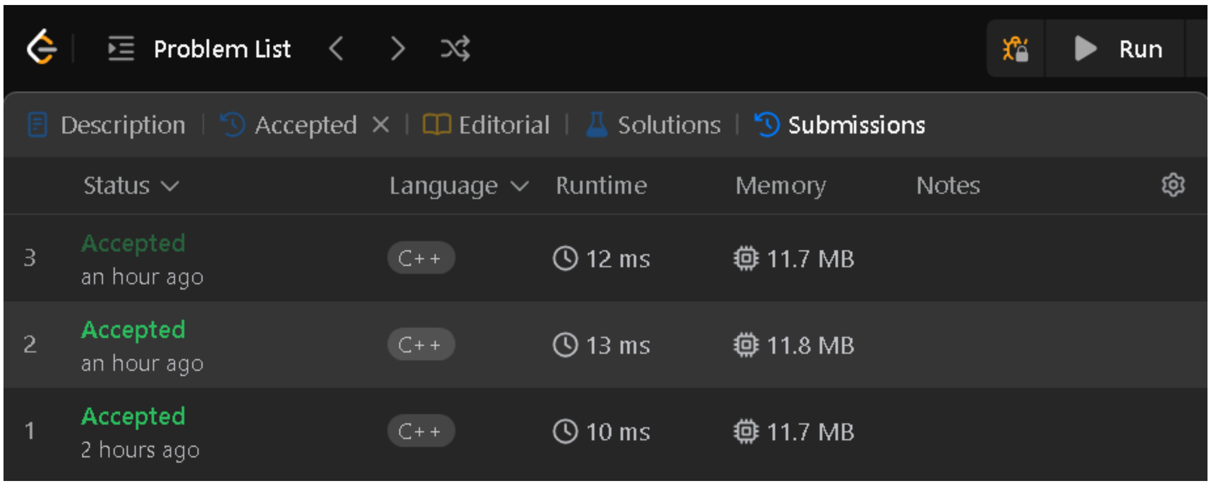
Description | Accepted | Editorial | Solutions | Submissions

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

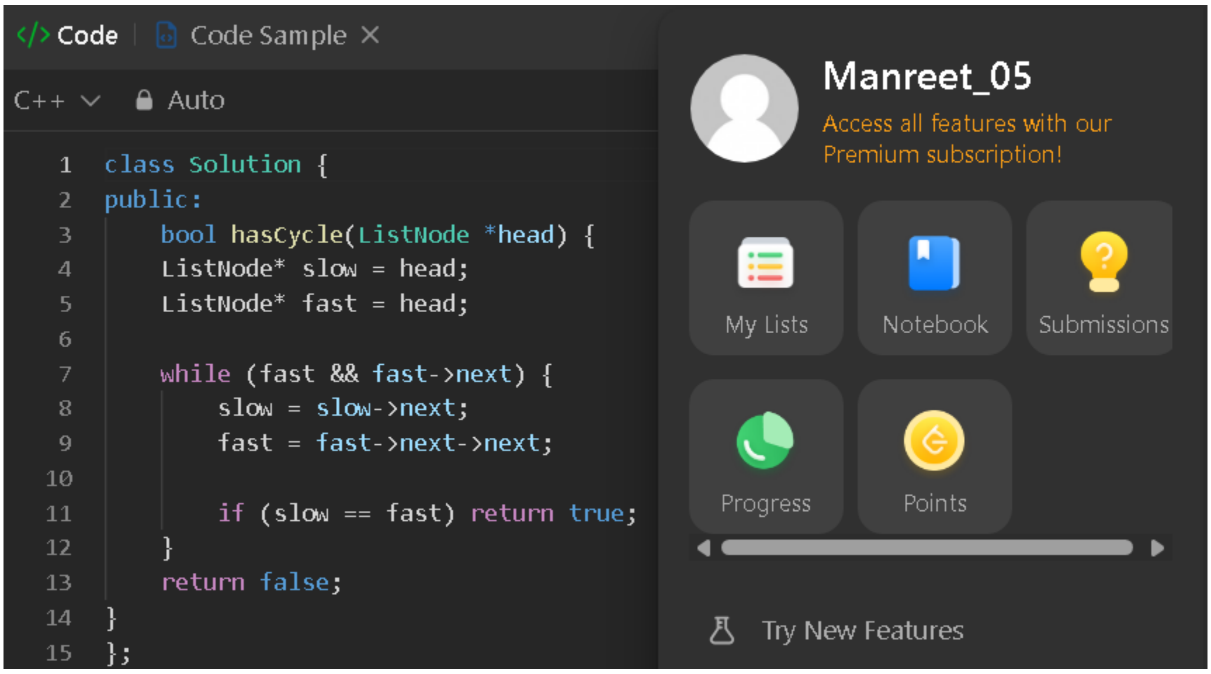
C++ Auto
Manreet_05
Access all features with our Premium subscription!
My Lists
Notebook
Submissions
Progress
Points
Try New Features
Orders

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2)
    {
    if (!list1) return list2;
    if (!list2) return list1;

    if (list1->val < list2->val) {
        list1->next = mergeTwoLists(list1->next, list2);
        return list1;
    } else {
        list2->next = mergeTwoLists(list1, list2->next);
        return list2;
    }
}
};
```

6. Detect a cycle in a linked list:

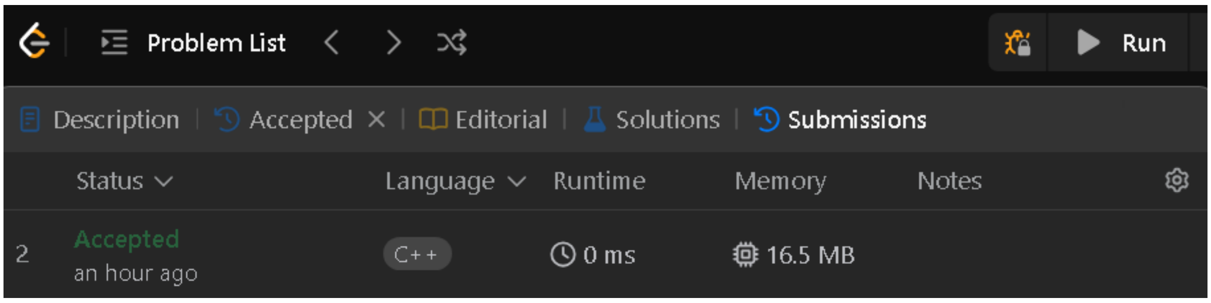


	Status	Language	Runtime	Memory	Notes
3	Accepted an hour ago	C++	12 ms	11.7 MB	
2	Accepted an hour ago	C++	13 ms	11.8 MB	
1	Accepted 2 hours ago	C++	10 ms	11.7 MB	

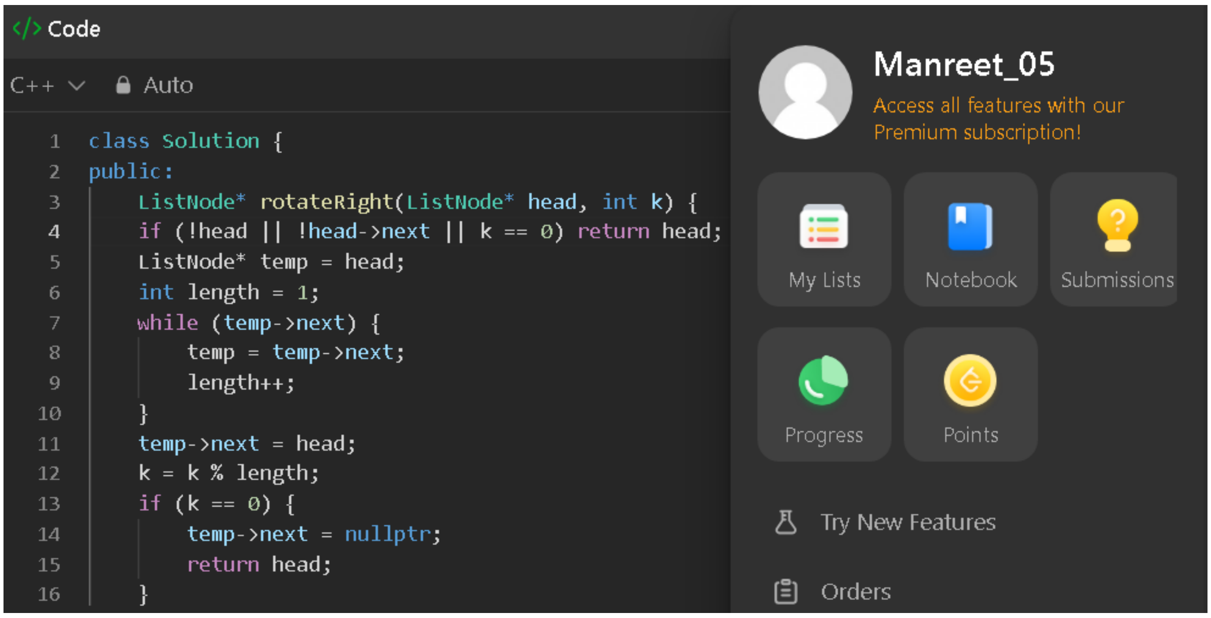


```cpp
class Solution {
public:
    bool hasCycle(ListNode *head) {
    ListNode* slow = head;
    ListNode* fast = head;

    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;

        if (slow == fast) return true;
    }
    return false;
 }
};
```

7.Rotate a list:

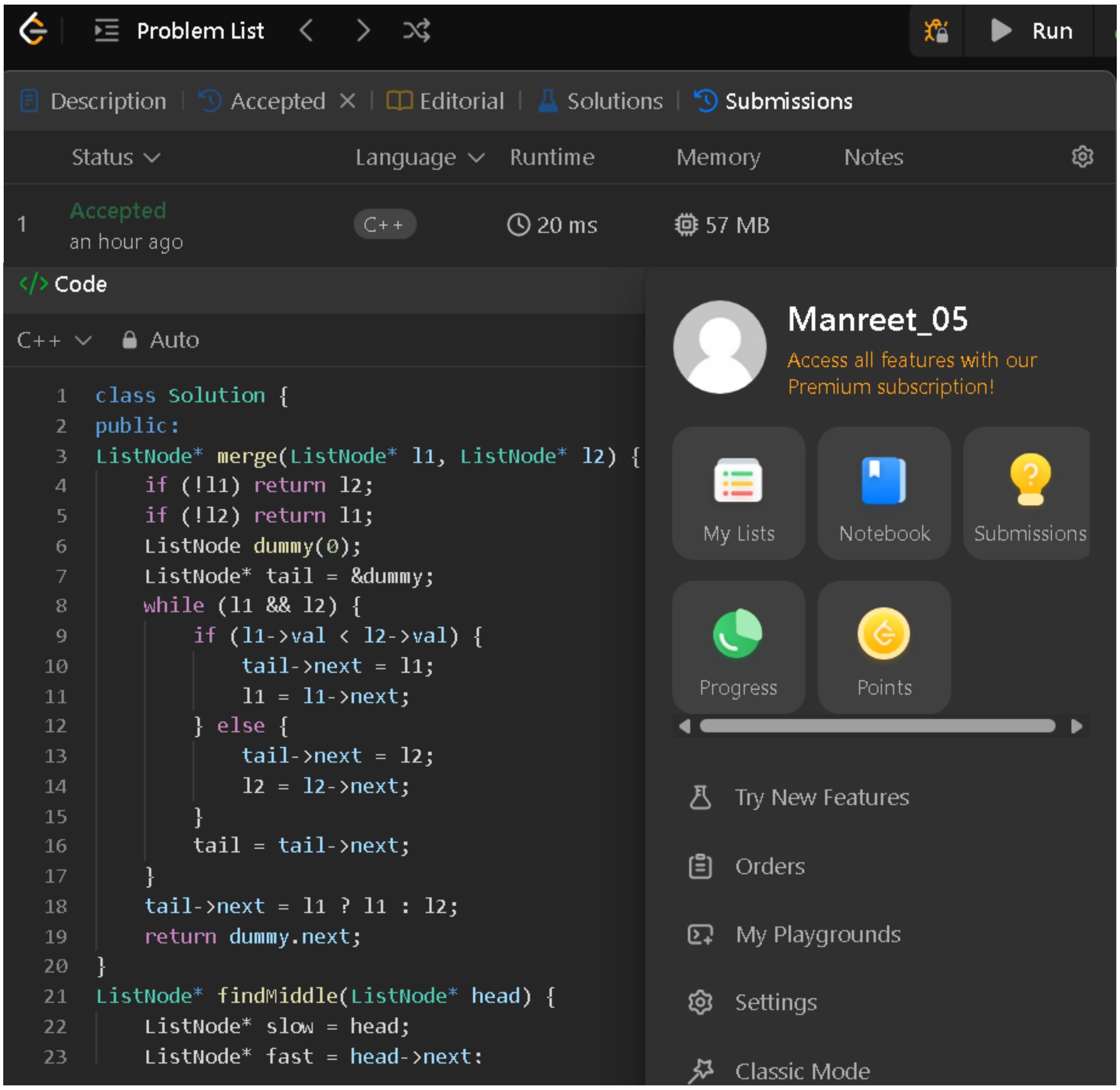


	Status	Language	Runtime	Memory	Notes
2	Accepted an hour ago	C++	0 ms	16.5 MB	



```cpp
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;
    }
```

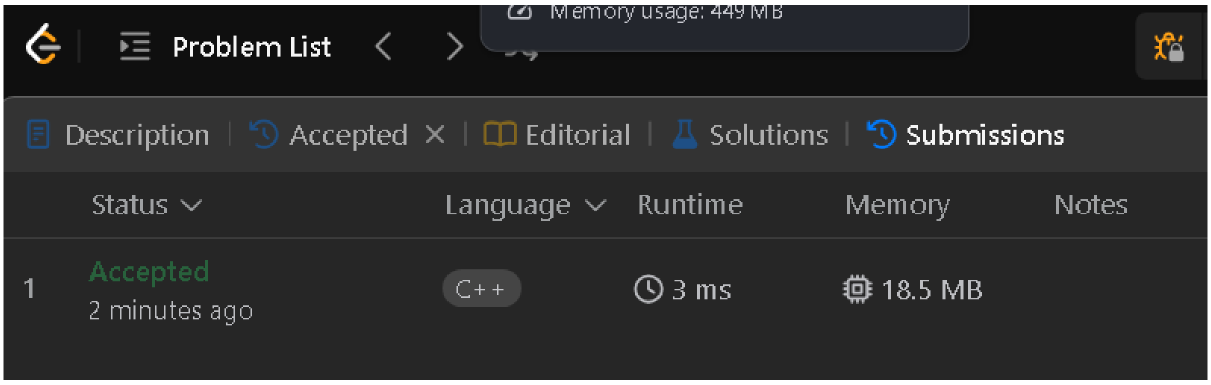
8. Sort List:



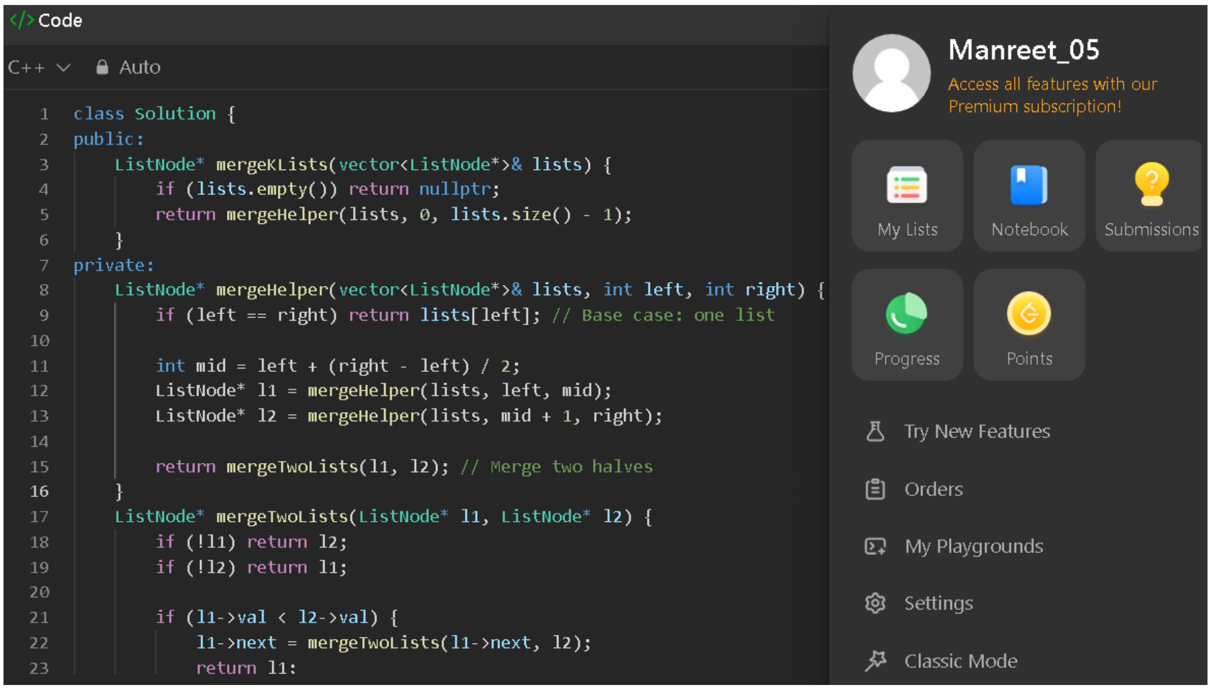
	Status	Language	Runtime	Memory	Notes
1	Accepted an hour ago	C++	20 ms	57 MB	

```cpp
class Solution {
public:
ListNode* merge(ListNode* l1, ListNode* l2) {
    if (!l1) return l2;
    if (!l2) return l1;
    ListNode dummy(0);
    ListNode* tail = &dummy;
    while (l1 && l2) {
        if (l1->val < l2->val) {
            tail->next = l1;
            l1 = l1->next;
        } else {
            tail->next = l2;
            l2 = l2->next;
        }
        tail = tail->next;
    }
    tail->next = l1 ? l1 : l2;
    return dummy.next;
}
ListNode* findMiddle(ListNode* head) {
    ListNode* slow = head;
    ListNode* fast = head->next;
```

9. Merge k sorted lists:



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



```cpp
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) return nullptr;
        return mergeHelper(lists, 0, lists.size() - 1);
    }
private:
    ListNode* mergeHelper(vector<ListNode*>& lists, int left, int right) {
        if (left == right) return lists[left]; // Base case: one list

        int mid = left + (right - left) / 2;
        ListNode* l1 = mergeHelper(lists, left, mid);
        ListNode* l2 = mergeHelper(lists, mid + 1, right);

        return mergeTwoLists(l1, l2); // Merge two halves
    }
    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:
```