

# ASSIGNMENT - 3

Student Name: Manreet Kaur

UID: 22BCS15550

Branch: BE-CSE

Section/Group: 608/B

Semester: 6<sup>th</sup>

Subject Name: AP LAB

## 1. Print Linked List:

90% Refund

Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

≡

<> Problem

📖 Editorial

Submissions

💬 Comments

My Submissions

All Submissions

🔄 Refresh

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:

Problem List < > 🔍

Description

Accepted ×

Editorial

Solutions

Submissions

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

The screenshot shows a code editor with a dark theme. At the top, there are buttons for 'Run' and 'Submit'. The code is written in C++ and defines a class 'Solution' with a public method 'deleteDuplicates'. The method takes a 'ListNode\* head' and returns a 'ListNode\*'. It uses a 'while' loop to traverse the list, comparing the value of the current node with the next node. If they are equal, the next node is skipped. Otherwise, the current node's 'next' pointer is updated to the next node's 'next' pointer. The method returns the head of the modified list.

```
1 class Solution {
2 public:
3     ListNode* deleteDuplicates(ListNode* head) {
4         ListNode* current = head;
5         while (current && current->next) {
6             if (current->val == current->next->val) {
7                 current->next = current->next->next;
8             } else {
9                 current = current->next;
10            }
11        }
12        return head;
13    }
14 };
15
```

On the right side, there is a user profile for 'Manreet\_05' with a prompt to 'Access all features with our Premium subscription!'. Below the profile are icons for 'My Lists', 'Notebook', 'Submissions', 'Progress', and 'Points'. At the bottom, there is a 'Try New Features' button.

3. Reverse a linked list:

The screenshot shows a code editor with a dark theme. At the top, there are buttons for 'Run' and 'Submit'. The code is written in C++ and defines a class 'Solution' with a public method 'reverseList'. The method takes a 'ListNode\* head' and returns a 'ListNode\*'. It uses a 'while' loop to traverse the list, reversing the 'next' pointers of each node. The method returns the new head of the reversed list.

```
1 class Solution {
2 public:
3     ListNode* reverseList(ListNode* head) {
4         ListNode* prev = nullptr;
5         ListNode* current = head;
6         while (current) {
7             ListNode* nextNode = current->next;
8             current->next = prev;
9             prev = current;
10            current = nextNode;
11        }
12        return prev;
13    }
14 };
15
```

On the right side, there is a user profile for 'Manreet\_05' with a prompt to 'Access all features with our Premium subscription!'. Below the profile are icons for 'My Lists', 'Notebook', 'Submissions', 'Progress', and 'Points'. At the bottom, there is a 'Try New Features' button.

#### 4. Delete middle node of a list:

The screenshot shows a LeetCode submission interface. At the top, there's a navigation bar with 'Problem List', 'Description', 'Editorial', 'Solutions', and 'Submissions'. Below this is a table with columns: Status, Language, Runtime, Memory, and Notes. The first submission (ID 1) is 'Accepted', written in C++, with a runtime of 0 ms and memory usage of 312 MB. The code is displayed in a dark-themed editor. On the right, there's a user profile for 'Manreet\_05' with a premium subscription badge and several interactive buttons like 'My Lists', 'Notebook', 'Submissions', 'Progress', and 'Points'.

```
1 class Solution {
2 public:
3     ListNode* deleteMiddle(ListNode* head) {
4         if (!head || !head->next) return nullptr;
5
6         ListNode *slow = head, *fast = head, *prev = nullptr;
7
8         while (fast && fast->next) {
9             prev = slow;
10            slow = slow->next;
11            fast = fast->next->next;
12        }
13        prev->next = slow->next;
14        return head;
15    }
16};
```

#### 5. Merge two sorted linked lists:

The screenshot shows a LeetCode submission interface for the 'Merge Two Sorted Linked Lists' problem. The submission table shows two entries, both 'Accepted' in C++ with 0 ms runtime and around 19 MB memory. The code is shown in a dark-themed editor. The right sidebar features the same user profile 'Manreet\_05' and navigation buttons as the previous screenshot.

```
1 class Solution {
2 public:
3     ListNode* mergeTwoLists(ListNode* list1, ListNode* list2)
4     {
5         if (!list1) return list2;
6         if (!list2) return list1;
7
8         if (list1->val < list2->val) {
9             list1->next = mergeTwoLists(list1->next, list2);
10            return list1;
11        } else {
12            list2->next = mergeTwoLists(list1, list2->next);
13            return list2;
14        }
15    }
16};
```

## 6. Detect a cycle in a linked list:

The screenshot displays a coding platform interface. At the top, there's a navigation bar with 'Problem List', navigation arrows, and a 'Run' button. Below this is a tabbed interface with 'Description', 'Accepted', 'Editorial', 'Solutions', and 'Submissions'. The 'Submissions' tab is active, showing a table of submissions.

	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	

Below the table, the 'Code' tab is selected, showing a C++ code sample. The code implements a function `hasCycle` using Floyd's Cycle-Finding algorithm (slow and fast pointers).

```
1 class Solution {
2 public:
3     bool hasCycle(ListNode *head) {
4         ListNode* slow = head;
5         ListNode* fast = head;
6
7         while (fast && fast->next) {
8             slow = slow->next;
9             fast = fast->next->next;
10
11             if (slow == fast) return true;
12         }
13         return false;
14     }
15 };
```

On the right side, there's a user profile for 'Manreet\_05' with a premium subscription notice. Below the profile are buttons for 'My Lists', 'Notebook', 'Submissions', 'Progress', and 'Points'. At the bottom, there's a 'Try New Features' button.

## 7. Rotate a list:

The screenshot displays a coding platform interface, similar to the one above. The 'Submissions' tab is active, showing a table of submissions.

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

Code

C++ Auto

```

1 class Solution {
2 public:
3     ListNode* rotateRight(ListNode* head, int k) {
4         if (!head || !head->next || k == 0) return head;
5         ListNode* temp = head;
6         int length = 1;
7         while (temp->next) {
8             temp = temp->next;
9             length++;
10        }
11        temp->next = head;
12        k = k % length;
13        if (k == 0) {
14            temp->next = nullptr;
15            return head;
16        }

```

Manreet\_05

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features

Orders

## 8. Sort List:

Problem List

< > ↺

Run

Description Accepted Editorial Solutions Submissions

Status

Language

Runtime

Memory

Notes

1	Accepted an hour ago	C++	20 ms	57 MB	
---	-------------------------	-----	-------	-------	--

Code

C++ Auto

```

1 class Solution {
2 public:
3     ListNode* merge(ListNode* l1, ListNode* l2) {
4         if (!l1) return l2;
5         if (!l2) return l1;
6         ListNode dummy(0);
7         ListNode* tail = &dummy;
8         while (l1 && l2) {
9             if (l1->val < l2->val) {
10                tail->next = l1;
11                l1 = l1->next;
12            } else {
13                tail->next = l2;
14                l2 = l2->next;
15            }
16            tail = tail->next;
17        }
18        tail->next = l1 ? l1 : l2;
19        return dummy.next;
20    }
21    ListNode* findMiddle(ListNode* head) {
22        ListNode* slow = head;
23        ListNode* fast = head->next;

```

Manreet\_05

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

## 9. Merge k sorted lists:

Memory Usage: 449 MB

Problem List

Description | Accepted × | Editorial | Solutions | Submissions

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

Code

C++ ▾ Auto

```
1 class Solution {
2 public:
3     ListNode* mergeKLists(vector<ListNode*>& lists) {
4         if (lists.empty()) return nullptr;
5         return mergeHelper(lists, 0, lists.size() - 1);
6     }
7 private:
8     ListNode* mergeHelper(vector<ListNode*>& lists, int left, int right) {
9         if (left == right) return lists[left]; // Base case: one list
10
11         int mid = left + (right - left) / 2;
12         ListNode* l1 = mergeHelper(lists, left, mid);
13         ListNode* l2 = mergeHelper(lists, mid + 1, right);
14
15         return mergeTwoLists(l1, l2); // Merge two halves
16     }
17     ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
18         if (!l1) return l2;
19         if (!l2) return l1;
20
21         if (l1->val < l2->val) {
22             l1->next = mergeTwoLists(l1->next, l2);
23             return l1;
24         }
```

**Manreet\_05**  
Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode