

# ASSIGNMENT

**Student Name: Sandhya Bharti**

**UID: 22BCS11412**

**Branch: BE-CSE**

**Section/Group: 608/B**

**Semester: 6<sup>th</sup>**

**Subject Name: AP LAB**

## 1. Print Linked List

**My Submissions**

All Submissions

 Refresh

Time (IST)	Status	Marks	Lang	Test Cases	Code
2025-02-21 10:58:06	Correct	1	cpp	1112 / 1112	<a href="#">View</a>

```
class Solution {
public:
    // Function to display the elements of a linked list in the same line
    void printList(Node *head) {
        Node* temp = head;
        while (temp) {
            cout << temp->data;
            if(temp->next) cout<< " "; // Print data followed by space
            temp = temp->next;
        }
        // Newline at the end
    }
};
// } Driver Code Ends
```

## 2. Remove duplicates from a sorted list:

Description   Editorial   Solutions   Submissions				
Status	Language	Runtime	Memory	Notes
1 <b>Accepted</b> a day ago	C++	0 ms	16.1 MB	

```
Code
C++ Auto
7 * ListNode(int x) : val(x), next(nullptr) {}
8 * ListNode(int x, ListNode *next) : val(x), next(next) {}
9 * };
10 */
11
12 class Solution {
13 public:
14     ListNode* deleteDuplicates(ListNode* head) {
15         ListNode* current = head;
16         while (current && current->next) {
17             if (current->val == current->next->val) {
18                 ListNode* duplicate = current->next;
19                 current->next = current->next->next;
20                 delete duplicate;
21             } else {
22                 current = current->next;
23             }
24         }
25         return head;
26     }
27 };

```

**sandhya**  
Access all features with our Premium subscription!

- My Lists
- Notebook
- Submissions
- Progress
- Points
- Try New Features
- Orders
- My Playgrounds
- Settings
- Classic Mode

## 3. Reverse a linked list:

Description   Accepted   Editorial   Solutions   Submissions				
Status	Language	Runtime	Memory	Notes
1 <b>Accepted</b> in 3 minutes	C++	0 ms	13.3 MB	

```
</> Code
C++ v Auto
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* reverseList(ListNode* head) {
14         ListNode* prev = nullptr;
15         ListNode* current = head;
16         while (current) {
17             ListNode* nextNode = current->next;
18             current->next = prev;
19             prev = current;
20             current = nextNode;
21         }
22         return prev;
23     }
24 };

```

S sandhya

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features

Orders

My Playgrounds

4. Delete middle node of a list:

Description   Editorial   Solutions   Submissions					
Status		Language	Runtime	Memory	Notes
1	Accepted 11 minutes ago	C++	1 ms	312.2 MB	

```
</> Code
C++ v Auto
10  /
11 class Solution {
12 public:
13     ListNode* deleteMiddle(ListNode* head) {
14         if (!head || !head->next) return nullptr;
15
16         ListNode* slow = head, *fast = head, *prev = nullptr;
17
18         while (fast && fast->next) {
19             prev = slow;
20             slow = slow->next;
21             fast = fast->next->next;
22         }
23
24         prev->next = slow->next;
25         delete slow;
26         return head;
27     }
28 };
29
30

```

S sandhya

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

## 5. Merge two sorted linked lists:

	Description	Editorial	Solutions	Submissions	
	Status	Language	Runtime	Memory	Notes
1	Accepted Feb 27, 2025	C++	0 ms	19.6 MB	

```
</> Code
C++ v Auto
10  */
11  class Solution {
12  public:
13      ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
14          ListNode dummy(0);
15          ListNode* tail = &dummy;
16
17          while (list1 && list2) {
18              if (list1->val < list2->val) {
19                  tail->next = list1;
20                  list1 = list1->next;
21              } else {
22                  tail->next = list2;
23                  list2 = list2->next;
24              }
25              tail = tail->next;
26          }
27
28          if (list1) {
29              tail->next = list1;
30          } else {
31              tail->next = list2;
32          }
33
34          return dummy.next;
35      }
36  };

```

S sandhya

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

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

	Description	Accepted	Editorial	Solutions	Submissions	
	Status	Language	Runtime	Memory	Notes	
1	Accepted in 3 minutes	C++	8 ms	11.9 MB		

Code

C++ ▾ 🔒 Auto

```
7  * };
8  */
9  class Solution {
10 public:
11     bool hasCycle(ListNode *head) {
12         ListNode *slow = head, *fast = head;
13
14         while (fast && fast->next) {
15             slow = slow->next;
16             fast = fast->next->next;
17
18             if (slow == fast) return true;
19         }
20         return false;
21     }
22 }
```

S sandhya

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features •

Orders

7. Rotate a list:

Description   Accepted ×   Editorial   Solutions   Submissions				
Status ▾	Language ▾	Runtime	Memory	Notes
1 Accepted in 3 minutes	C++	0 ms	16.5 MB	

Code

C++ ▾ 🔒 Auto

```
11 class Solution {
12 public:
13     ListNode* rotateRight(ListNode* head, int k) {
14         if (!head || !head->next || k == 0) return head;
15
16         ListNode* temp = head;
17         int length = 1;
18         while (temp->next) {
19             temp = temp->next;
20             length++;
21         }
22
23         temp->next = head;
24         k = k % length;
25         int newTailPos = length - k - 1;
26         temp = head;
27         for (int i = 0; i < newTailPos; i++) {
28             temp = temp->next;
29         }
30         head = temp->next;
31         temp->next = nullptr;
32
33         return head;
34     }
35 }
```

S sandhya

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features •

Orders

My Playgrounds

Settings

Classic Mode

## 8. Sort List:

[Description](#) | [Accepted](#) × | [Editorial](#) | [Solutions](#) | [Submissions](#) ⌵ ⌵

	Status <span>⌵</span>	Language <span>⌵</span>	Runtime	Memory	Notes
1	<b>Accepted</b> in 3 minutes	C++	18 ms	57.2 MB	

</> Code

C++ ⌵ 🔒 Auto

```
11 class Solution {
12 public:
13     ListNode* sortList(ListNode* head) {
14         if (!head || !head->next) return head;
15         ListNode* mid = getMid(head);
16         ListNode* left = sortList(head);
17         ListNode* right = sortList(mid);
18         return merge(left, right);
19     }
20
21 private:
22     ListNode* getMid(ListNode* head) {
23         ListNode* slow = head, *fast = head->next;
24         while (fast && fast->next) {
25             slow = slow->next;
26             fast = fast->next->next;
27         }
28         ListNode* mid = slow->next;
29         slow->next = nullptr; // Split list
30         return mid;
31     }
32
33     ListNode* merge(ListNode* l1, ListNode* l2) {
34         ListNode dummy(0), *tail = &dummy;
35         while (l1 && l2) {
36             if (l1->val < l2->val) swap(l1, l2);
37             tail->next = l2;
38             l2 = l2->next;
39             tail = tail->next;
40         }
41         tail->next = l1;
42         return dummy->next;
43     }
44 }
```

S sandhya

Access all features with our Premium subscription!

My Lists

Notebook

Submissions

Progress

Points

Try New Features •

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

## 9. Merge k sorted lists:

[Description](#) | [Accepted](#) × | [Editorial](#) | [Solutions](#) | [Submissions](#) ⌵ ⌵

	Status <span>⌵</span>	Language <span>⌵</span>	Runtime	Memory	Notes
1	<b>Accepted</b> in 3 minutes	C++	5 ms	18.3 MB	

Code

C++ Auto

```
10  */
11  class Solution {
12  public:
13      ListNode* mergeKLists(vector<ListNode*>& lists) {
14          auto cmp = [](ListNode* a, ListNode* b) { return a->val >
15              priority_queue<ListNode*, vector<ListNode*>, decltype(cmp)
16
17          for (auto list : lists)
18              if (list) pq.push(list);
19
20          ListNode dummy(0), *tail = &dummy;
21
22          while (!pq.empty()) {
23              ListNode* node = pq.top(); pq.pop();
24              tail->next = node;
25              tail = tail->next;
26              if (node->next) pq.push(node->next);
27          }
28          return dummy.next;
29      }
30  };
```

S

sandhya

Access all features with our  
Premium subscription!



My Lists



Notebook



Submissions



Progress



Points



Try New Features



Orders



My Playgrounds



Settings



Classic Mode