

ASSIGNMENT-3

NAME: Khushi Raj Shekhar

UID: 22BCS17009

SUBJECT: AP-2


CLASS- IOT-610/B

1. Print Linked

List: <https://www.geeksforgeeks.org/problems/print-linked-list-elements/0>

```
Python3 Start Timer
1  #Your task is to complete this function
2  #Your function should print the data in one line only
3
4  '''
5  class Node:
6      def __init__(self, x):
7          self.data = x
8          self.next = None
9  '''
10 class Solution:
11     # Function to display the elements of a linked list in same line
12     def printList(self, node):
13         #code here
14         while node:
15             print(node.data,end=" ")
16             node=node.next
17
18
19 # } Driver Code Ends
```

Compilation ResultsCustom InputY.O.G.I. (AI Bot)

Problem Solved Successfully 

[Suggest Feedback](#)

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
2 / 7
Accuracy : 28%

Time Taken
0.42

2. Remove duplicates from a sorted list: <https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/>

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* curr = head;

        while (curr != nullptr) {
            while (curr->next && curr->val == curr->next->val)
                curr->next = curr->next->next;
            curr = curr->next;
        }

        return head;
    }
};
```

Editorial Solutions Submissions				
Status ▾	Language ▾	Runtime	Memory	Notes
1 Accepted Feb 21, 2025	C++	🕒 0 ms	🧠 16.3 MB	

3. Reverse a linked

list: <https://leetcode.com/problems/reverse-linked-list/description/>

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;

        while (head != nullptr) {
            ListNode* next = head->next;
            head->next = prev;
            prev = head;
            head = next;
        }

        return prev;
    }
};
```

Description | Accepted ×

← All Submissions

Accepted 28 / 28 testcases passed
submitted at Mar 05, 2025 10:06

Editorial Solution

Runtime 0 ms | Beats 100.00% 🌿
Analyze Complexity

Memory 13.56 MB | Beats 18.51%

150%
100%

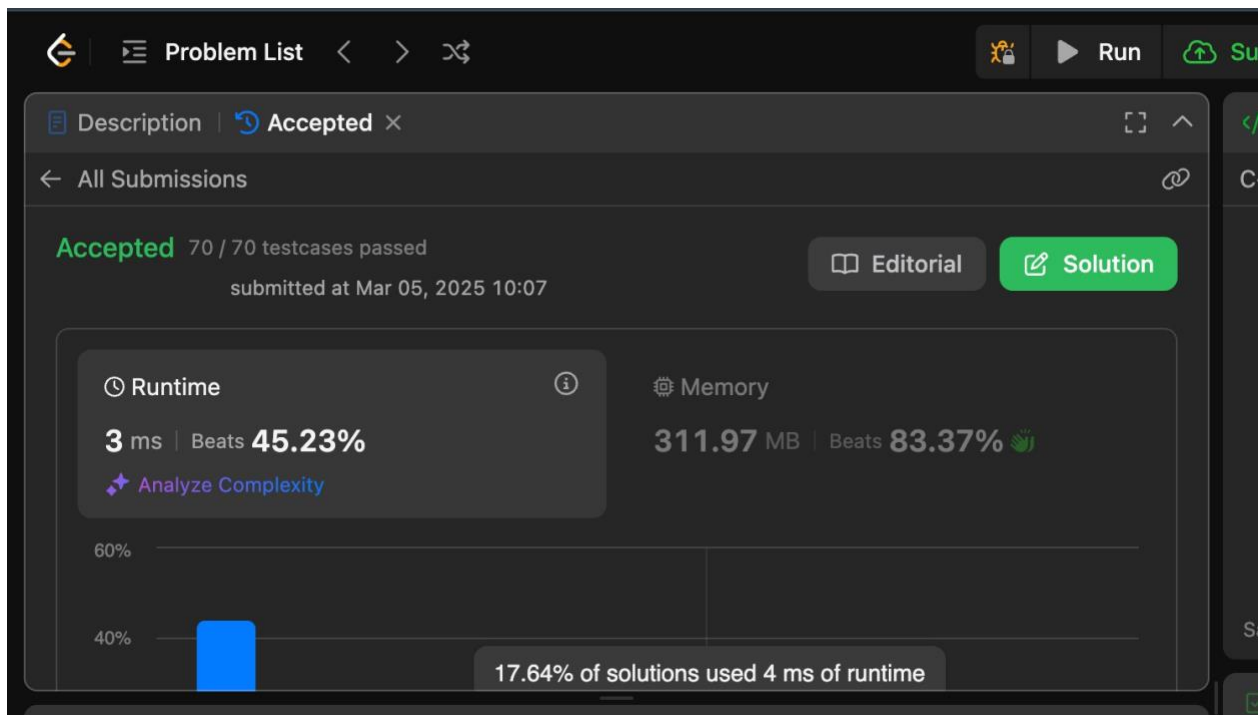
Editorial Solutions Submissions

4. Delete middle node of a list: <https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/>

```
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        ListNode dummy(0, head);
        ListNode* slow = &dummy;
        ListNode* fast = &dummy;

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

        // Delete the middle node.
        slow->next = slow->next->next;
        return dummy.next;
    }
};
```



5. Merge two sorted linked

lists: <https://leetcode.com/problems/merge-two-sorted-lists/description/>

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if (!list1 || !list2)
            return list1 ? list1 : list2;
        if (list1->val > list2->val)
            swap(list1, list2);
        list1->next = mergeTwoLists(list1->next, list2);
        return list1;
    }
};
```

Editorial Solutions Submissions					
Status ▾	Language ▾	Runtime	Memory	Notes	⚙
1 Accepted Feb 18, 2025	C++	0 ms	19.4 MB		

6. Detect a cycle in a linked

list: <https://leetcode.com/problems/linked-list-cycle/description/>

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

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

        return false;
    }
};
```

Editorial Solutions Submissions				
Status ▾	Language ▾	Runtime	Memory	Notes
1 Accepted Feb 26, 2025	C++	🕒 10 ms	💾 11.9 MB	

7. Rotate a list: <https://leetcode.com/problems/rotate-list/description/>

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0)
            return head;

        ListNode* tail;
        int length = 1;
        for (tail = head; tail->next; tail = tail->next)
            ++length;
        tail->next = head; // Circle the list.

        const int t = length - k % length;
        for (int i = 0; i < t; ++i)
            tail = tail->next;
        ListNode* newHead = tail->next;
        tail->next = nullptr;
    }
};
```

```

        return newHead;
    }
};

```

Problem List < > 🔍

Description | Accepted ×

← All Submissions 🔗

Accepted 232 / 232 testcases passed
submitted at Mar 05, 2025 10:11

Editorial Solution

⌚ Runtime ⓘ 0 ms | Beats 100.00% 🌟
🔗 Analyze Complexity

⚙️ Memory 16.28 MB | Beats 93.87% 🌟

100%
50%

Editorial Solutions Submissions

8. Sort List: <https://leetcode.com/problems/sort-list/description/>

```

class Solution {
public:
    ListNode* sortList(ListNode* head) {
        const int length = getLength(head);
        ListNode dummy(0, head);

        for (int k = 1; k < length; k *= 2) {
            ListNode* curr = dummy.next;
            ListNode* tail = &dummy;

```



```

        while (curr != nullptr) {
            ListNode* l = curr;
            ListNode* r = split(l, k);
            curr = split(r, k);
            auto [mergedHead, mergedTail] = merge(l, r);
            tail->next = mergedHead;
            tail = mergedTail;
        }
    }

    return dummy.next;
}

private:
int getLength(ListNode* head) {
    int length = 0;
    for (ListNode* curr = head; curr; curr = curr->next)
        ++length;
    return length;
}

ListNode* split(ListNode* head, int k) {
    while (--k && head)
        head = head->next;
    ListNode* rest = head ? head->next : nullptr;
    if (head != nullptr)
        head->next = nullptr;
    return rest;
}

pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {
    ListNode dummy(0);
    ListNode* tail = &dummy;

    while (l1 && l2) {
        if (l1->val > l2->val)
            swap(l1, l2);
        tail->next = l1;
        l1 = l1->next;
        tail = tail->next;
    }
    tail->next = l1 ? l1 : l2;
    while (tail->next != nullptr)

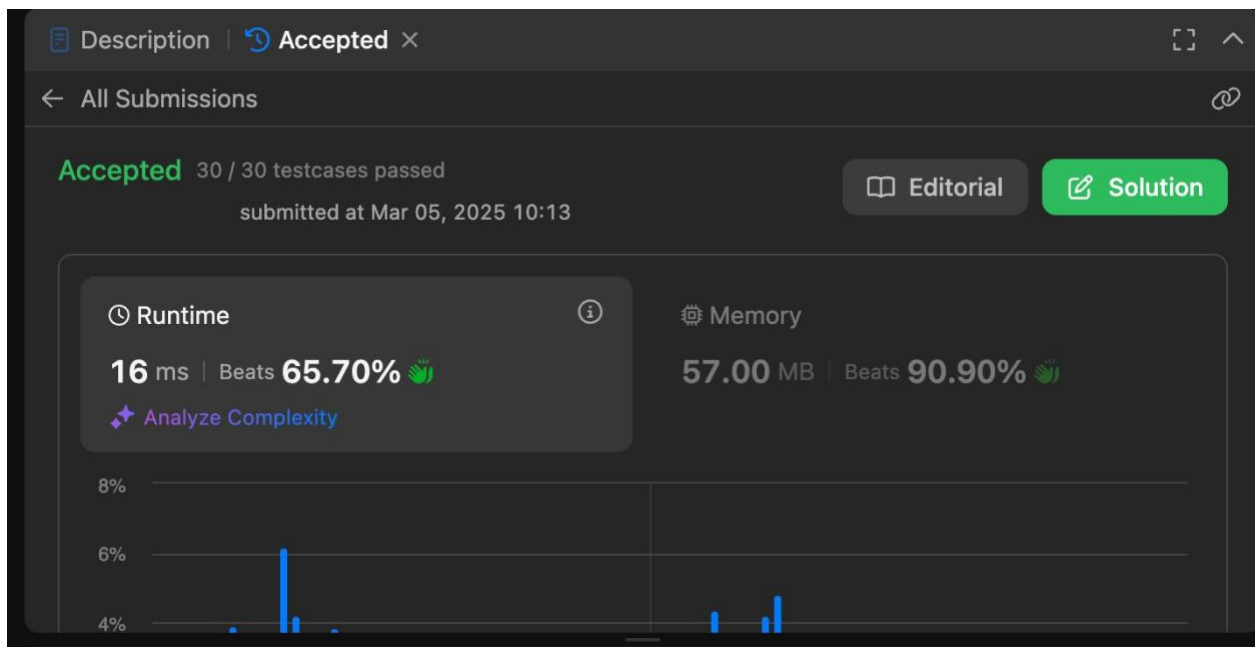
```

```

        tail = tail->next;

        return {dummy.next, tail};
    }
};

```



9. Merge k sorted

lists: <https://leetcode.com/problems/merge-k-sorted-lists/description/>

```

class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        ListNode dummy(0);
        ListNode* curr = &dummy;
        auto compare = [](ListNode* a, ListNode* b) { return a->val > b->val; };
        priority_queue<ListNode*, vector<ListNode*>, decltype(compare)> minHeap(
            compare);
    }
};

```

```
for (ListNode* list : lists)
    if (list != nullptr)
        minHeap.push(list);

while (!minHeap.empty()) {
    ListNode* minNode = minHeap.top();
    minHeap.pop();
    if (minNode->next)
        minHeap.push(minNode->next);
    curr->next = minNode;
    curr = curr->next;
}

return dummy.next;
}
```

Description | Accepted ×

← All Submissions [Link](#)

Accepted 134 / 134 testcases passed
submitted at Mar 05, 2025 10:14

[Editorial](#) [Solution](#)

🕒 Runtime ⓘ

2 ms | Beats **70.60%** 🌿

🔗 [Analyze Complexity](#)

💾 Memory

18.34 MB | Beats **80.73%** 🌿

75%
50%