

Harsh Pandey

22BCS17030

IOT-609/B

Advance Programming

Assignment-3

1. Print Linked List

CODE:

```
class Solution {
public:
    // Function to display the elements of a linked list in same line
    void printList(Node *head) {
        while (head!= NULL)
        {
            cout<<head->data<<" ";
            head = head->next;
        }
        // your code goes here
    }
};
```

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

[Suggest Feedback](#)

Test Cases Passed

1112 / 1112

Attempts : Correct / Total

3 / 3

Accuracy : 100%

Time Taken

0.1

2. Remove duplicates from a sorted list:

CODE:

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if(!head) return nullptr;
```

```

ListNode* current = head;
while (current && current->next){
    if(current->val == current->next->val){
        ListNode* temp = current->next;
        current->next= current->next->next;
        delete temp;
    }
    else{
        current = current->next;
    }
}
return head;
}
};

```

The screenshot shows a C++ IDE with a code editor and a test results panel. The code in the editor implements a function to delete duplicates from a linked list. The test results panel shows that the code was accepted with a runtime of 0 ms. The input was a linked list with values [1, 1, 2] and the output was [1, 2].

```

0  * ListNode() : val(0), next(nullptr) {}
7  * ListNode(int x) : val(x), next(nullptr) {}
8  * ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 /*
11 class Solution {
12 public:
13     ListNode* deleteDuplicates(ListNode* head) {
14         if(!head) return nullptr;
15         ListNode* current = head;
16         while (current && current->next){
17             if(current->val == current->next->val){

```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =

[1,1,2]

Output

[1,2]

3. Reverse a linked list:

CODE:

```

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

```

```

        current=nextNode;
    }
    return prev;
}
};

```

The screenshot shows a C++ IDE interface. On the left, a 'Submissions' table lists two entries: an 'Accepted' submission from Feb 25, 2025, and a 'Compile Error' submission from Dec 26, 2024. The main editor displays C++ code for reversing a linked list. The code includes a `ListNode` struct and a `reverseList` function. Below the code, the 'Test Result' section shows an 'Accepted' status with a runtime of 0 ms. It details 'Case 1' with an input list `[1,2,3,4,5]` and an output list `[5,4,3,2,1]`.

Status	Language	Runtime	Memory
Accepted	C++	0 ms	13.4
Compile Error	C++	N/A	N/A

```

4  *   int val;
5  *   ListNode *next;
6  *   ListNode() : val(0), next(nullptr) {}
7  *   ListNode(int x) : val(x), next(nullptr) {}
8  *   ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* reverseList(ListNode* head) {

```

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

head =
[1,2,3,4,5]

Output

[5,4,3,2,1]

Expected

4. Delete middle node of a list:

CODE:

```

class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if(!head || !head->next) return nullptr;
        ListNode* slow = head;
        ListNode* fast = head;
        ListNode* prev = nullptr;
        while(fast && fast->next){
            prev=slow;
            slow=slow->next;
            fast=fast->next->next;
        }
        prev->next=slow->next;
        delete slow;
        return head; }
};

```

The screenshot displays a coding platform interface. On the left, a table lists submissions with columns for Status, Language, Runtime, and Memory. The submissions show a mix of 'Accepted' and 'Runtime Error' statuses. On the right, the 'Code' editor shows C++ code for a linked list problem. The 'Testcase' section shows 'Accepted' with a runtime of 0 ms. The input is 'head = [1,3,4,7,1,2,6]' and the output is '[1,3,4,1,2,6]'. The expected output is also shown as '[1,3,4,1,2,6]'.

5. Merge two sorted linked lists:

CODE:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        ListNode dummy(0); // Dummy node to simplify code
        ListNode* tail = &dummy; // Pointer to build the merged list

        while (list1 && list2) {
            if (list1->val <= list2->val) {
                tail->next = list1;
                list1 = list1->next;
            } else {
                tail->next = list2;
                list2 = list2->next;
            }
            tail = tail->next; // Move the tail pointer
        }

        // Attach the remaining elements of the non-empty list
        tail->next = list1 ? list1 : list2;

        return dummy.next; // Return the merged list (skip dummy node)
    }
};
```

Submissions

Status	Language	Runtime	Memory
Accepted	C++	0 ms	19.4
Compile Error	C++	N/A	N/A

```

10  */
11  class Solution {
12  public:
13      ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
14          ListNode dummy(0); // Dummy node to simplify code
15          ListNode* tail = &dummy; // Pointer to build the merged list
16
17          while (list1 && list2) {

```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

list1 =
[1,2,4]

list2 =
[1,3,4]

Output

[1,1,2,3,4,4]

6. Detect a cycle in a linked list:

CODE:

```

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

```

Submissions

Status	Language	Runtime	Memory
Accepted	C++	3 ms	11.8

```

1  /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode(int x) : val(x), next(NULL) {}
7  * };

```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =
[3,2,0,-4]

pos =
1

Output

true

7. Rotate a list:

CODE:

```
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;
        int stepsToNewHead = length - k;
        ListNode* newTail = head;
        for (int i = 1; i < stepsToNewHead; i++) {
            newTail = newTail->next;
        }
        head = newTail->next;
        newTail->next = nullptr;
        return head;
    }
};
```

The screenshot displays a code editor interface for a C++ solution. The left sidebar shows the 'Problem List' and 'All Submissions' tabs. The main editor area displays the C++ code for the 'rotateRight' function. Below the code, the 'Runtime' and 'Memory' performance metrics are shown. The 'Runtime' section indicates 0 ms execution time and 100.00% beats. The 'Memory' section shows 16.30 MB usage and 93.87% beats. A bar chart visualizes the performance across different test cases. The right sidebar shows the 'Testcase' and 'Test Result' tabs. The 'Test Result' tab displays the input and output for a specific test case, showing that the solution is 'Accepted'.

Runtime: 0 ms | Beats 100.00%

Memory: 16.30 MB | Beats 93.87%

Testcase: Case 1

Input: head = [1,2,3,4,5], k = 2

Output: [4,5,1,2,3]

Expected: [4,5,1,2,3]

8. Sort List:

CODE:

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) return head; // Base case

        // Step 1: Split the list into two halves
        ListNode* mid = getMid(head);
        ListNode* left = head;
        ListNode* right = mid->next;
        mid->next = nullptr; // Break the list into two halves

        // Step 2: Recursively sort each half
        left = sortList(left);
        right = sortList(right);

        // Step 3: Merge sorted halves
        return merge(left, right);
    }

private:
    // Function to find the middle node using the slow-fast pointer approach
    ListNode* getMid(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head->next;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
        }
        return slow;
    }

    // Function to merge two sorted linked lists
    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;

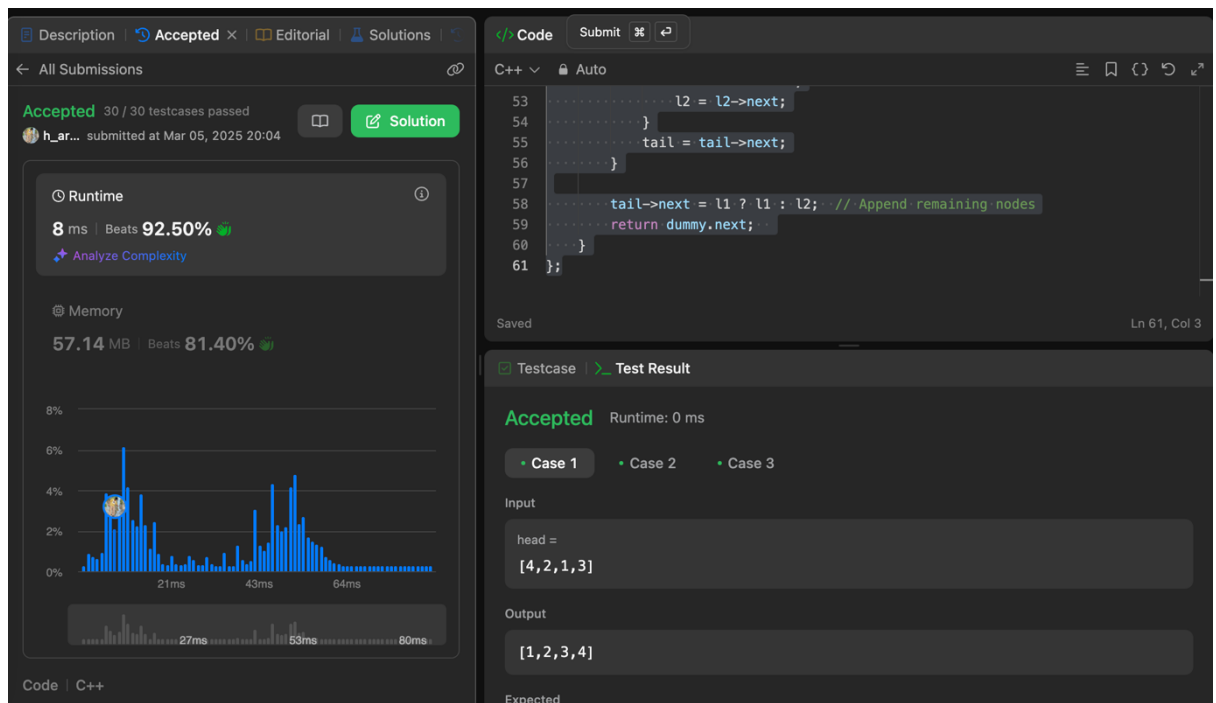
        while (l1 && l2) {
            if (l1->val < l2->val) {
```

```

        tail->next = l1;
        l1 = l1->next;
    } else {
        tail->next = l2;
        l2 = l2->next;
    }
    tail = tail->next;
}

tail->next = l1 ? l1 : l2; // Append remaining nodes
return dummy.next;
}
};

```



9. Merge k sorted lists:

CODE:

```

class Solution {
public:
    struct Compare {
        bool operator()(ListNode* a, ListNode* b) {
            return a->val > b->val;
        }
    };
};

ListNode* mergeKLists(vector<ListNode*>& lists) {
    priority_queue<ListNode*, vector<ListNode*>, Compare> minHeap;

```



```

for (auto list : lists) {
    if (list) minHeap.push(list);
}
ListNode dummy;
ListNode* tail = &dummy;
while (!minHeap.empty()) {
    ListNode* smallest = minHeap.top();
    minHeap.pop();
    tail->next = smallest;
    tail = tail->next;
    if (smallest->next) {
        minHeap.push(smallest->next);
    }
}
return dummy.next;
}
};

```

The screenshot displays a coding platform interface with the following components:

- Problem List:** Shows the problem is "Accepted" with 134 / 134 testcases passed. The user "h_ar..." submitted the solution on Mar 05, 2025 at 20:09.
- Runtime and Memory:**
 - Runtime: 0 ms, Beats 100.00%.
 - Memory: 18.44 MB, Beats 66.01%.
 - A bar chart shows the user's performance compared to other submissions across different runtime ranges (1ms, 63ms, 125ms, 187ms).
- Code Editor:** Displays the C++ code for the solution, including a custom comparator for a min-heap.
- Test Result:**
 - Overall status: **Accepted** (Runtime: 0 ms).
 - Testcase details for Case 1:
 - Input: `lists = [[1,4,5], [1,3,4], [2,6]]`
 - Output: `[1,1,2,3,4,4,5,6]`
 - Expected: `[1,1,2,3,4,4,5,6]`