AP ASSIGNMENT-2

Name- Manya Goyal **UID-** 22BCS16483

1. **Print Linked List:**

C++ (g++ 5.4) Start Timer

```cpp
// } Driver Code Ends



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

// } Driver Code Ends
```

Problem Solved Successfully

Suggest Feedback

Test Cases Passed	Attempts : Correct / Total
1112 / 1112	**2 / 4**
	Accuracy : 50%

2. **Remove duplicates from a sorted list:**

</> Code

C++ Auto

```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;
    }
};
```

Ln 14, Col 3 Saved

Testcase | >_ Test Result

Accepted Runtime: 0 ms

Accepted 168 / 168 testcases passed

manya13 submitted at Mar 05, 2025 20:29

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

16.29 MB | Beats 35.16%

3. **Reverse a linked list:**

C++ Auto

```cpp
class Solution {
public:
    ListNode* reverseList(ListNode* head) {

        ListNode* prev = NULL;
        ListNode* curr = head;

        while(curr != NULL){
            ListNode* forward = curr->next;
            curr->next = prev;
            prev = curr;
            curr = forward;

        }
        return prev;
    }
```

Ln 18, Col 3 Saved

Run

Testcase | >_ Test Result

Accepted 28 / 28 testcases passed

manya13 submitted at Mar 05, 2025 20:31

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

13.34 MB | Beats 70.13%

150%

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

C++ Auto

```cpp
public:
    ListNode* deleteMiddle(ListNode* head) {
        if(!head->next) return NULL;
        if(!head->next->next){
            head->next = NULL;
            return head;
        }
        ListNode* slow = head;
        ListNode* fast = head;
        while(fast && fast->next){
            slow = slow->next;
            fast = fast->next->next;
        }
        slow->val = slow->next->val;
        slow->next = slow->next->next;
        return head;
    }
};
```

Ln 19, Col 3 | Saved

Testcase | Test Result

Accepted 70 / 70 testcases passed

manya13 submitted at Mar 05, 2025 20:32

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

312.00 MB | Beats 83.37%

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

C++ Auto

```cpp
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if(list1 == NULL || list2 == NULL){
            return list1 == NULL ? list2 : list1;
        }

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

Ln 18, Col 3 | Saved

Run

Testcase | Test Result

Accepted 208 / 208 testcases passed

manya13 submitted at Mar 05, 2025 20:33

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

19.52 MB | Beats 28.41%

100%

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

C++ Auto

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

Ln 23, Col 3 | Saved

Run

Testcase | Test Result

Accepted Runtime: 0 ms

Accepted 29 / 29 testcases passed

manya13 submitted at Mar 05, 2025 20:36

Editorial Solution

Runtime

6 ms | Beats 90.79%

Analyze Complexity

Memory

11.71 MB | Beats 79.92%

7. **Rotate a list:**

C++ Auto

```cpp
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0) return head;
        ListNode* tail = head;
        int length = 1;

        while (tail->next) {
            tail = tail->next;
            length++;
        }
        k %= length;
        if (k == 0) return head;
        tail->next = head;
        int stepsToNewHead = length - k;
        ListNode* newTail = tail;
        while (stepsToNewHead--) {
            newTail = newTail->next;
        }
        ListNode* newHead = newTail->next;
        newTail->next = nullptr;
        return newHead;
    }
};
```

Ln 21, Col 33 | Saved

Run

Testcase | Test Result

Accepted Runtime: 0 ms

Accepted 232 / 232 testcases passed

manya13 submitted at Mar 05, 2025 20:38

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

16.28 MB | Beats 93.87%

8. **Sort List:**

```
public:
ListNode* merge(ListNode* list1, ListNode* list2) {
        ListNode* C = new ListNode(100) ;
        ListNode* temp = C ;
        while(list1!= NULL && list2!=NULL){
            if(list1->val <= list2->val){
                temp->next = list1 ;
                list1 = list1->next ;
                temp = temp ->next ;
            }
            else{
                temp->next = list2 ;
                list2 = list2->next ;
                temp = temp->next ;
            }
        }
        if(list1 == NULL )temp->next = list2 ;
        if(list2==NULL) temp->next = list1 ;
    return C->next ;
    }
    ListNode* sortList(ListNode* head) {
        if(head==NULL || head->next == NULL)return head ;
        // to find the middle of the linked list
        ListNode* slow = head ;
        ListNode* fast = head ;
        while(fast->next!=NULL && fast->next->next!=NULL){
            slow = slow->next ;
            fast = fast->next->next ; }
        ListNode* a = head ;
        ListNode* b = slow->next ;
        slow->next  = NULL ;
        a = sortList(a) ;
        b = sortList(b) ;
        ListNode* c = merge(a,b) ;
        return c ;
    }
```

Accepted 30 / 30 testcases passed

manya13 submitted at Mar 05, 2025 20:41

Editorial Solution

Runtime

42 ms | Beats 48.60%

Analyze Complexity

Memory

75.83 MB | Beats 8.72%

9. **Merge k sorted lists:**

C++ Auto

```cpp
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) {
            return nullptr;
        }
        return mergeKListsHelper(lists, 0, lists.size() - 1);
    }
    ListNode* mergeKListsHelper(vector<ListNode*>& lists, int start, int end) {
        if (start == end) {
            return lists[start];
        }
        if (start + 1 == end) {
            return merge(lists[start], lists[end]);
        }
        int mid = start + (end - start) / 2;
        ListNode* left = mergeKListsHelper(lists, start, mid);
        ListNode* right = mergeKListsHelper(lists, mid + 1, end);
        return merge(left, right); }
    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode* dummy = new ListNode(0);
        ListNode* curr = dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                curr->next = l1;
                l1 = l1->next;
            } else {
                curr->next = l2;
                l2 = l2->next;
            }
            curr = curr->next;
        }
        curr->next = l1 ? l1 : l2;
        return dummy->next;
    }
```

Accepted 134 / 134 testcases passed

manya13 submitted at Mar 05, 2025 20:43

Editorial | Solution

Runtime

4 ms | Beats 48.89%

Analyze Complexity

Memory

23.90 MB | Beats 7.35%

75%