

AP Assignment-2

Ques 1

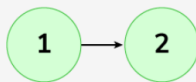
Print Linked List

Difficulty: **Basic** Accuracy: **60.71%** Submissions: **142K+** Points: **1**

Given a linked list. Print all the elements of the linked list separated by space followed.

Examples:

Input: LinkedList : 1 -> 2



Output: 1 2

Explanation: The linked list contains two elements 1 and 2. The elements are printed in a single line.

Input: Linked List : 49 -> 10 -> 30



Output Window

Compilation Results

Custom Input

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

Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed

1112 / 1112

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

1 / 1

Your Total Score: 63 ↑

Time Taken

0.1

Solve Next

[Count Linked List Nodes](#)

[Delete Alternate Nodes](#)

```
23 struct Node* next;  
24  
25 Node(int x) {  
26     data = x;  
27     next = nullptr;  
28 }  
29  
30 */  
31 /*  
32 Print elements of a linked list on console  
33 Head pointer input could be NULL as well for empty list  
34 */  
35  
36 class Solution {  
37 public:  
38     void printList(Node* head) {  
39         // Node* temp = head;  
40         // while(temp != NULL){  
41             cout<<temp->data<<" ";  
42             temp = temp->next;  
43         }  
44         while (head) {  
45             cout << head->data << " ";  
46             head = head->next;  
47         }  
48     }  
49 };  
50 // } Driver Code Ends
```



Custom Input

Compile & Run

Submit

Ques 2

Description

Editorial

Solutions

Submissions

83. Remove Duplicates from Sorted List

Solved

Copy

Copy Markdown

Easy

Topics

Companies

Given the **head** of a sorted linked list, *delete all duplicates such that each element appears only once*. Return the linked list **sorted** as well.

Example 1:

9.1K 108 83 Online

Code

Accepted

All Submissions

Accepted 168 / 168 testcases passed

Pratham Sanan submitted at Aug 02, 2024 23:08

Editorial

Solution

Runtime

8 ms | Beats 1.28%

Analyze Complexity

Memory

16.79 MB | Beats 11.11%

100%

50%

Testcase

Test Result

```
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if(head == NULL){
            return NULL;
        }

        ListNode* curr = head;

        while(curr != NULL){

            if((curr -> next != NULL) && (curr -> val == curr -> nex
                ListNode* next_next = curr -> next -> next;
                ListNode* nodeToDelete = curr -> next;
                delete(nodeToDelete);
                curr -> next = next_next;
            }
            else{
                curr = curr -> next;
            }
        }
        return head;
    }
}
```

Ques 3

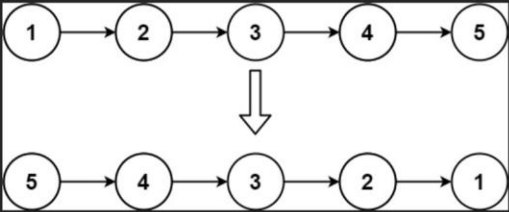
Description | Editorial | Solutions | Submissions

206. Reverse Linked List Copy Copy Markdown Solved ✓

Easy Topics Companies

Given the `head` of a singly linked list, reverse the list, and return the reversed list.

Example 1:



```
graph LR; 1((1)) --> 2((2)); 2 --> 3((3)); 3 --> 4((4)); 4 --> 5((5)); 5 --> 4; 4 --> 3; 3 --> 2; 2 --> 1;
```

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

22.6K 272 326 Online

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

Code Accepted

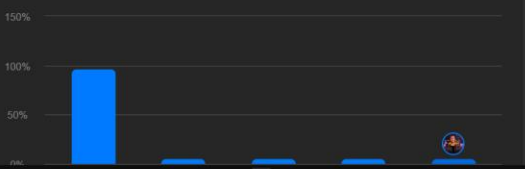
All Submissions

Accepted 28 / 28 testcases passed
Pratham Sanan submitted at Jul 27, 2024 18:15

Editorial Solution

Runtime
4 ms | Beats 1.52%
[Analyze Complexity](#)

Memory
12.99 MB | Beats 99.99%



Testcase Test Result

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        if(head == NULL || head -> next == NULL){
            return head;
        }

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

        while(curr != NULL){
            frwd = curr -> next;
            curr -> next = prev;
            prev = curr;
            curr = frwd;
        }
        return prev;
    }
};
```

Ques 5

Description

Editorial

Solutions

Submissions

21. Merge Two Sorted Lists

Copy

Copy Markdown

Solved

Easy

Topics

Companies

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

Example 1:

23K

419

458 Online

Code

Accepted

All Submissions

Accepted

208 / 208 testcases passed

Pratham Sanan submitted at Aug 04, 2024 18:19

Editorial

Solution

Runtime

7 ms | Beats 1.27%

Analyze Complexity

Memory

19.82 MB | Beats 5.26%

100%

50%

Testcase

Test Result

```

ListNode* solve(ListNode* list1, ListNode* list2){

    if(list1 -> next == NULL){
        list1 -> next = list2;
        return list1;
    }

    ListNode* curr1 = list1;
    ListNode* next1 = curr1 -> next;
    ListNode* curr2 = list2;
    ListNode* next2 = curr2 -> next;

    while(next1 != NULL && curr2 != NULL){
        if((curr2 -> val >= curr1 -> val) && (curr2 -> val <= next1 -> val)){
            curr1 -> next = curr2;
            next2 = curr2 -> next;
            curr2 -> next = next1;

            curr1 = curr2;
            curr2 = next2;
        }
        else{
            // curr1 aur next1 ko aage badana hai
            curr1 = next1;
            next1 = next1 -> next;
        }
    }

    if(next1 == NULL){
        curr1 -> next = curr2;
        return list1;
    }
    return list1;
}

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

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

```

Ques 6

Description | Editorial | Solutions | Submissions

141. Linked List Cycle

Copy Copy Markdown Solved

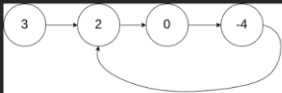
Easy Topics Companies

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter.**

Return `true` if there is a cycle in the linked list. Otherwise, return `false`.

Example 1:



```
graph LR; 3((3)) --> 2((2)); 2 --> 0((0)); 0 --> -4((-4)); -4 --> 2
```

Input: `head = [3,2,0,-4], pos = 1`
Output: `true`
Explanation: There is a cycle in the linked list, where the tail node's next pointer is connected to the node at index 1.

16.2K 360 207 Online

Code Accepted

All Submissions

Accepted 29 / 29 testcases passed

Pratham Sanan submitted at Jul 31, 2024 23:26

Editorial Solution

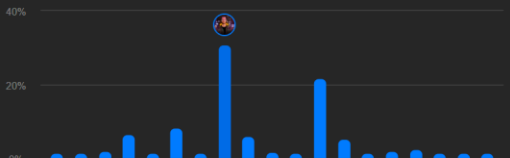
Runtime

8 ms | Beats 80.86%

Analyze Complexity

Memory

10.83 MB | Beats 99.95%



Testcase Test Result

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if(head == NULL){
            return NULL;
        }

        ListNode* slow = head;
        ListNode* fast = head;

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

            slow = slow->next;

            if(slow == fast){
                return slow;
            }
        }

        return NULL;
    }
};
```

Ques 7

Description | Editorial | Solutions | Submissions

61. Rotate List

Copy Copy Markdown

Solved

Medium Topics Companies

Given the `head` of a linked list, rotate the list to the right by `k` places.

Example 1:

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

Example 2:

10.2K 104 98 Online

Code Accepted

All Submissions

Accepted 232 / 232 testcases passed

Pratham Sanan submitted at Sep 14, 2024 10:49

Editorial Solution

Runtime

3 ms | Beats 3.74%

Analyze Complexity

Memory

16.69 MB | Beats 4.05%

100%

50%

0%

Testcase Test Result

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {

        if (head == NULL || head->next == NULL) {
            return head;
        }

        int lgth=1;
        ListNode* temp = head;
        while(temp->next){
            temp = temp->next;
            lgth++;
        }

        temp->next = head;

        k = k%lgth;
        int stepsToNewHead = lgth-k;
        ListNode* newTail = head;
        for(int i=1;i<stepsToNewHead;i++){
            newTail = newTail->next;
        }

        ListNode* newHead = newTail->next;
        newTail->next = NULL;

        return newHead;
    }
};
```

Ques 8

Description

Editorial

Solutions

Submissions

148. Sort List

Copy

Copy Markdown

Solved

Medium

Topics

Companies

Given the `head` of a linked list, return the list after sorting it in **ascending order**.

Example 1:

Input: head = [4,2,1,3]
Output: [1,2,3,4]

Example 2:

12.2K 111 91 Online

Code

Accepted

All Submissions

Accepted

30 / 30 testcases passed

Pratham Sanan submitted at Aug 10, 2024 15:31

Editorial

Solution

Runtime

159 ms Beats: 5.04%

Analyze Complexity

Memory

74.96 MB Beats: 48.73%

Code

C++

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 */

```

Testcase

Test Result

```

ListNode* findMid(ListNode* head){
    ListNode* slow = head;
    ListNode* fast = head->next;

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

```

```

ListNode* merge(ListNode* left, ListNode* right){
    if(left == NULL){
        return right;
    }
    if(right == NULL){
        return left;
    }
}

```

```

ListNode* ans = new ListNode(-1);
ListNode* temp = ans;

```

```

while(left != NULL && right != NULL){
    if(left->val < right->val){
        temp -> next=left;
        temp = left;
        left = left->next;
    }
    else{
        temp->next = right;
        temp = right;
        right = right->next;
    }
}
}

```

```

while(left != NULL){
    temp -> next=left;
    temp = left;
    left = left->next;
}

```

```

while(right != NULL){
    temp->next = right;
    temp = right;
    right = right->next;
}

```

```

ans = ans->next;
return ans;
}

```

```

class Solution {
public:
    ListNode* sortList(ListNode* head) {
        //base case
        if(head == NULL || head->next == NULL){
            return head;
        }

        ListNode* mid = findMid(head);

        ListNode* left = head;
        ListNode* right = mid->next;
        mid->next = NULL;

        left = sortList(left);
        right = sortList(right);

        ListNode* result = merge(left, right);

        return result;
    }
};

```