

NAME: ADITYA CHAURASIA SUBJECT: ADVANCED PROGRAMMING

UID: 22BCS11655

SECTION: IOT_612/B

90% Refund
Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

Problem Editor Submissions Comments

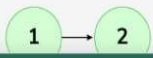
Print Linked List

Difficulty: Basic Accuracy: 60.71% Submissions: 141K+ Points: 1

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

Examples:

Input: LinkedList : 1 -> 2



Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Expected Output:
1 2

Output Difference
1 2

Ask Yogi Bot

```
1 // Driver Code Ends
2
3 struct Node {
4     int data;
5     struct Node* next;
6 }
7
8 Node(int x) {
9     data = x;
10    next = nullptr;
11 }
12
13
14 // Print elements of a linked list on console
15 // Head pointer input could be NULL as well for empty list
16
17 class Solution {
18 public:
19     // Function to display the elements of a linked list in same line
20     void printList(Node* head) {
21         // your code goes here
22     }
23 };
24
25
```

Problem List < > ⌕

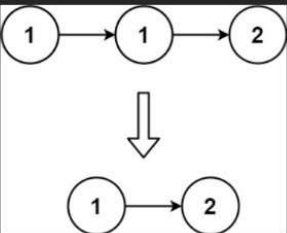
Description Editorial Solutions Submissions

83. Remove Duplicates from Sorted List

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:



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

```
1 class Solution {
2 public:
3     ListNode* deleteDuplicates(ListNode* head) {
4         if (!head) return nullptr;
5         ListNode* current = head;
6         while (current && current->next) {
7             if (current->val == current->next->val) {
8                 ListNode* temp = current->next;
9                 current->next = current->next->next;
10                delete temp;
11            } else {
12                current = current->next;
13            }
14        }
15        return head;
16    }
17 }
```

Saved

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =
[1,1,2]

Output

[1,2]

9.1K 106 84 Online

Problem List

RunSubmit

0

DescriptionAcceptedEditorialSolutionsSubmissions

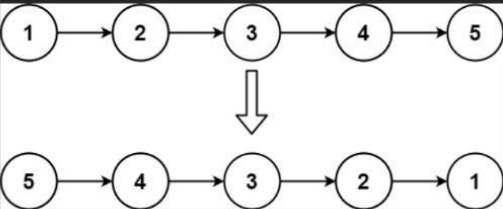
206. Reverse Linked List

Solved

EasyTopicsCompanies


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

Example 1:



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

Example 2:



22.5K265371 Online

CodeTestcase

C++Auto

```
6 while (current) {
7     ListNode* nextNode = current->next; // Store next node
8     current->next = prev; // Reverse the link
9     prev = current; // Move prev to current
10    current = nextNode; // Move current to next
11 }
12 return prev; // New head of the reversed list
13 }
14 }
15
```

Saved

Test Result

AcceptedRuntime: 0 ms

Case 1Case 2Case 3

Input

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

Output

[5,4,3,2,1]

Problem List

RunSubmit

0

DescriptionEditorialSolutionsSubmissions

2095. Delete the Middle Node of a Linked List


MediumTopicsCompaniesHint

You are given the head of a linked list. Delete the middle node, and return the head of the modified linked list.

The middle node of a linked list of size n is the $\lfloor n / 2 \rfloor^{\text{th}}$ node from the start using 0-based indexing, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

For n = 1, 2, 3, 4, and 5, the middle nodes are 0, 1, 1, 2, and 2, respectively.

Example 1:



Input: head = [1,3,4,7,1,2,6]
Output: [1,3,4,1,2,6]
Explanation:
The above figure represents the given linked list. The indices of the nodes are written below.
Since n = 7, node 3 with value 7 is the middle node, which is marked in red. We return the new list after removing this node.

4.4K7167 Online

CodeTestcase

C++Auto

```
13 fast = fast->next->next;
14 }
15
16 prev->next = slow->next; // Delete the middle node
17 delete slow;
18
19 return head;
20
```

Saved

Test Result

AcceptedRuntime: 0 ms

Case 1Case 2Case 3

Input

head =
[1,3,4,7,1,2,6]

Output

[1,3,4,1,2,6]

Expected

[1,3,4,1,2,6]

Problem List

Run

Submit

0

Description

Editorial

Solutions

Submissions

21. Merge Two Sorted Lists

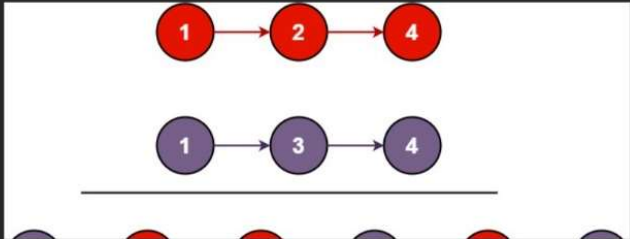
EasyTopicsCompanies

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:



22.9K411480 Online

Code

Testcase

C++Auto

```
23
24     ListNode* mergedHead = dummy->next;
25     delete dummy;
26     return mergedHead;
27 }
28 };
29
```

Saved

Test Result

AcceptedRuntime: 0 ms

Case 1Case 2Case 3

Input

list1 =

[1,2,4]

list2 =

[1,3,4]

Output

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

Problem List

Run

Submit

0

Description

Editorial

Solutions

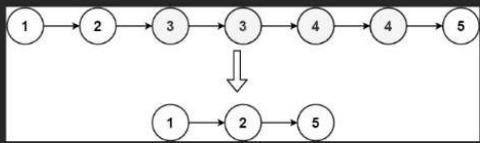
Submissions

82. Remove Duplicates from Sorted List II

MediumTopicsCompanies

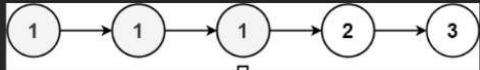
Given the `head` of a sorted linked list, *delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list.* Return the linked list **sorted** as well.

Example 1:



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

Example 2:



9.1K8262 Online

Code

Testcase

C++Auto

```
24     ListNode* newHead = dummy->next;
25     delete dummy;
26     return newHead;
27 }
28 };
29
```

Saved

Test Result

AcceptedRuntime: 0 ms

Case 1Case 2

Input

head =

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

Output

[1,2,5]

Expected

[1,2,5]

Problem List

16.2K

352

199 Online

141. Linked List Cycle

Easy

Topics

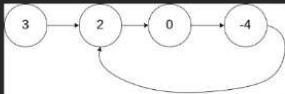
Companies

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.**

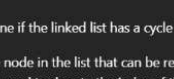
Return `true` if there is a cycle in the linked list. Otherwise, return `false`.

Example 1:



Input: `head = [3,2,0,-4]`, `pos = 1`
Output: `true`
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

Example 2:



Input: `head = [1,2,3,4]`, `pos = 0`
Output: `false`
Explanation: No cycle is present in the linked list.

Code

Testcase

```
C++ Auto
1 // If there is a cycle, return true; otherwise, return false.
2
3 bool hasCycle(ListNode head) {
4     if (head == null) return false;
5     ListNode slow = head;
6     ListNode fast = head;
7     while (slow != null && fast != null) {
8         slow = slow->next;
9         fast = fast->next->next;
10    }
11    return slow == fast;
12 }
```

Saved

> Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

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

pos =
1

Output

true

Problem List

Run

Submit

0

Description

Editorial

Solutions

Submissions

61. Rotate List

MediumTopicsCompanies

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:

Code

Testcase

```
C++  
23  
24  
25 // Break the loop and set the new head  
26 head = temp->next;  
27 temp->next = nullptr;  
28  
29 return head;  
30  
31 }  
32
```

Saved

Test Result

Accepted

Runtime: 0 ms

Case 1Case 2

Input

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

k =
2

10.2K

101

108 Online

Problem List

Run

Submit

0

Description

Editorial

Solutions

Submissions

148. Sort List

MediumTopicsCompanies

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:

Code

Testcase

```
C++  
39 tail = tail->next;  
40 }  
41 tail->next = ll ? ll : 12;  
42 return dummy.next;  
43  
44 };  
45
```

Saved

Test Result

Accepted

Runtime: 0 ms

Case 1Case 2Case 3

Input

head =
[4,2,1,3]

Output

[1,2,3,4]

12.1K

108

134 Online

Problem List

RunSubmit

0

DescriptionEditorialSolutionsSubmissions

142. Linked List Cycle II

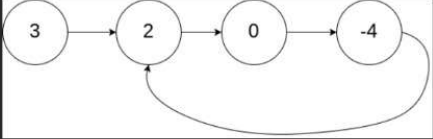
MediumTopicsCompanies

Given the `head` of a linked list, return *the node where the cycle begins*. If there is no cycle, return `null`.

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 (**0-indexed**). It is `-1` if there is no cycle. **Note that `pos` is not passed as a parameter**.

Do not modify the linked list.

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: tail connects to node index 1
Explanation: There is a cycle in the linked list, where tail connects to the second node.

14.1K179112 Online

CodeTestcase

C++Auto

20slow = slow->next;
21fast = fast->next;
22}
23return slow;
24}
25}
26return nullptr;
27}
28};
29

Saved

Test Result

AcceptedRuntime: 2 ms

Case 1Case 2Case 3

Input

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

pos =
1