

ASSIGNMENT 3

UID-22BCS10433

NAME-ARMAAN PREET SINGH

SECTION-606-B

1. Q1-Print Linked

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

The screenshot displays the GeeksforGeeks website interface for the 'Print Linked List' problem. The problem description states: 'Given a linked list. Print all the elements of the linked list separated by space followed.' Examples provided show an input linked list with nodes 1 and 2, resulting in the output '1 2'. The explanation notes that the elements are printed in a single line.

The code editor on the right shows a Java solution (Java 1.8) that defines a `Node` class and a `printList` method. The `printList` method iterates through the linked list and prints each node's data. The solution is as follows:

```
1 // Driver Code Ends
2
3 /* Node is defined as
4 class Node {
5     int data;
6     Node next;
7     Node(int x) {
8         data = x;
9         next = null;
10    }
11 }*/
12
13 Print elements of a linked list on console
14 Head pointer input could be NULL as well for empty list
15
16 class Solution {
17     // Function to display the elements of a linked list in same line
18     void printList(Node head) {
19         Node temp=head;
20         while(temp!=null)
21         {
22             System.out.print(temp.data+" ");
23             temp=temp.next;
24         }
25     }
26 }
27 }
```

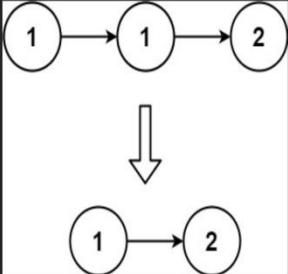
The 'Output Window' shows 'Compilation Results' for 'Custom Input' using 'Y.O.G.I. (AI Bot)'. The results indicate that the problem was solved successfully, with 1112/1112 test cases passed, 1/1 attempts correct, 100% accuracy, 1/1 points scored, and a time taken of 1.95 seconds.

Q2-Remove duplicates from a sorted
list: <https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/>

83. Remove Duplicates from Sorted List

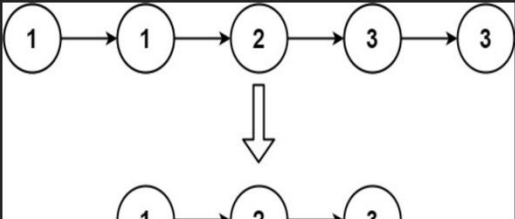
Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Example 1:



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

Example 2:



```
class Solution {  
    public ListNode deleteDuplicates(ListNode head) {  
        ListNode res = head;  
        while (head != null && head.next != null) {  
            if (head.val == head.next.val) {  
                head.next = head.next.next;  
            } else {  
                head = head.next;  
            }  
        }  
        return res;  
    }  
}
```

Testcase | Test Result

Case 1 Case 2 +

head =

[1,1,2]

90 Online

Q3. Reverse a linked list: <https://leetcode.com/problems/reverse-linked-list/description/>

206. Reverse Linked List

Easy Topics Companies

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:

Testcase 1 Test Result

Case 1 Case 2 Case 3 +

head =

[1,2,3,4,5]

```
public ListNode reverseList(ListNode head) {  
    ListNode prev = null;  
    ListNode curr = head;  
    ListNode next = null;  
  
    while (curr != null) {  
        next = curr.next;  
        curr.next = prev;  
        prev = curr;  
        curr = next;  
    }  
  
    return prev;  
}
```

Q4.Delete middle node of a list: <https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/>

Problem List

Run

Submit

🔍

📄

Description Editorial Solutions Accepted X Submissions

2095. Delete the Middle Node of a Linked List

Solved

Medium Topics Companies Hint

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.

Example 2:

Input: `head = [1,2,3,4]`
Output: `[1,2,4]`
Explanation:
The above figure represents the given linked list.
For $n = 4$, node 2 with value 3 is the middle node, which is marked in red.

4.5K 77 55 Online

Code

Java Auto

```
1 /**
2  * Definition for singly-linked list.
3  * public class ListNode {
4  *     int val;
5  *     ListNode next;
6  *     ListNode() {}
7  *     ListNode(int val) { this.val = val; }
8  *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
9  * }
10 */
11 class Solution {
12     public ListNode deleteMiddle(ListNode head) {
13         if(head.next == null){
14             return null;
15         }
16         ListNode slow = head;
17         ListNode fast = head.next;
18         while(fast.next != null && fast.next.next != null){
19             fast = fast.next.next;
20             slow = slow.next;
21         }
22         slow.next = slow.next.next;
23     }
24 }
```

Saved Ln 25, Col 2

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =

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

Output

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

Expected

Q5. Merge two sorted linked lists: <https://leetcode.com/problems/merge-two-sorted-lists/description/>

Problem List

21. Merge Two Sorted Lists

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:

Input: `list1 = [1,2,4]`, `list2 = [1,3,4]`
Output: `[1,1,2,3,4,4]`

Example 2:

Input: `list1 = []`, `list2 = []`
Output: `[]`

Example 3:

Input: `list1 = [1,2,4]`, `list2 = [1,3,4]`
Output: `[1,1,2,3,4,4]`

23K 420 539 Online

Code

Java

Auto

```
1 class Solution {
2     public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
3         ListNode dummy = new ListNode(0);
4         ListNode current = dummy;
5
6         while (list1 != null && list2 != null) {
7             if (list1.val < list2.val) {
8                 current.next = list1;
9                 list1 = list1.next;
10            } else {
11                current.next = list2;
12                list2 = list2.next;
13            }
14            current = current.next;
15        }
16
17        if (list1 != null) current.next = list1;
18        else current.next = list2;
19    }
20 }
21
```

Saved Ln 22, Col 29

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

list1 = [1,2,4]

list2 = [1,3,4]

Output

Q6. Detect a cycle in a linked list: <https://leetcode.com/problems/linked-list-cycle/description/>

Problem List

DescriptionEditorialSolutionsSubmissions

141. Linked List Cycle


EasyTopicsCompanies

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], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.

Example 3:

16.2K 360 267 Online

Code

JavaAuto

```
1 public class Solution {
2     public boolean hasCycle(ListNode head) {
3         ListNode fast = head;
4         ListNode slow = head;
5
6         while (fast != null && fast.next != null) {
7             fast = fast.next.next;
8             slow = slow.next;
9
10            if (fast == slow) {
11                return true;
12            }
13        }
14
15        return false;
16    }
17 }
```

SavedLn 17, Col 2

TestcaseTest Result

AcceptedRuntime: 0 ms

Case 1Case 2Case 3

Input

head =

[3,2,0,-4]

pos =

1

Output

Q7. Rotate a list: <https://leetcode.com/problems/rotate-list/description/>

Problem List

61. Rotate List

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:

Input: `head = [0,1,2]`, `k = 3`
Output: `[0,1,2]`

10.2K 104 108 Online

Code

Java

```
1 class Solution {
2     public ListNode rotateRight(ListNode head, int k) {
3         if(head == null || head.next == null){
4             return head;
5         }
6         int length = 1;
7         ListNode lastnode = head;
8         while(lastnode.next != null){
9             lastnode = lastnode.next;
10            length++;
11        }
12        lastnode.next = head;
13        ListNode subnodeprev = head;
14        int n = k%length;
15        for(int i=0; i< (length-n-1); i++){
16            subnodeprev = subnodeprev.next;
17        }
18        head = subnodeprev.next;
19        subnodeprev.next = null;
20        return head;
21    }
22 }
```

Saved Ln 13, Col 30

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =

[1,2,3,4,5]

k =

2

Output

Q8. Sort List: <https://leetcode.com/problems/sort-list/description/>

148. Sort List 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:

Input: `head = [-1,5,3,4,0]`
Output: `[-1,0,3,4,5]`

12.2K 111 123 Online

Code

```
1 class Solution {
2     public ListNode sortList(ListNode head) {
3         if(head==null){
4             return null;
5         }
6         ListNode ptr=head;
7         List<Integer> list=new ArrayList<>();
8         while(ptr!=null){
9             list.add(ptr.val);
10            ptr=ptr.next;
11        }
12        Collections.sort(list);
13        ListNode n = new ListNode(list.get(0));
14        head=n;
15        ListNode temp=head;
16        for(int i=1;i<list.size();i++){
17            ListNode n1 = new ListNode(list.get(i));
18            temp.next=n1;
19            temp=temp.next;
20        }
21        return head;
22    }
```

Saved Ln 23, Col 2

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =
[4,2,1,3]

Output

[1,2,3,4]

Q9. Merge k sorted lists: <https://leetcode.com/problems/merge-k-sorted-lists/description/>

The screenshot shows a LeetCode problem page for "Merge k sorted lists". The problem description states: "You are given an array of k linked-lists lists, each linked-list is sorted in ascending order. Merge all the linked-lists into one sorted linked-list and return it." Example 1 shows input lists = [[1,4,5],[1,3,4],[2,6]] and output [1,1,2,3,4,4,5,6]. Example 2 shows input lists = [] and output []. Constraints include k = lists.length, 0 <= k <= 10^4, 0 <= lists[i].length <= 500, -10^4 <= lists[i][j] <= 10^4, and lists[i] is sorted in ascending order. The code editor shows a Java solution using a recursive partition and merge function. The test result shows the solution is accepted with a runtime of 0 ms for Case 1.

Description | Editorial | Solutions | Accepted | Submissions

You are given an array of `k` linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:

Input: `lists = [[1,4,5],[1,3,4],[2,6]]`
Output: `[1,1,2,3,4,4,5,6]`
Explanation: The linked-lists are:
[
 1->4->5,
 1->3->4,
 2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6

Example 2:

Input: `lists = []`
Output: `[]`

Example 3:

Input: `lists = [[]]`
Output: `[]`

Constraints:

- `k == lists.length`
- `0 <= k <= 10^4`
- `0 <= lists[i].length <= 500`
- `-10^4 <= lists[i][j] <= 10^4`
- `lists[i]` is sorted in **ascending order**.

20.1K | 253 | 238 Online

Code

```
9  * }
10 */
11 class Solution {
12     public static ListNode mergeKLists(ListNode[] lists){
13         return partition(lists,0,lists.length-1);
14     }
15 }
16 public static ListNode partition(ListNode[] lists,int s,int e){
17     if(s==e) return lists[s];
18     if(s<e){
19         int q=(s+e)/2;
20         ListNode l1=partition(lists,s,q);
21         ListNode l2=partition(lists,q+1,e);
22         return merge(l1,l2);
23     }else
24         return null;
25 }
26
27 //This function is from Merge Two Sorted Lists.
28 public static ListNode merge(ListNode l1,ListNode l2){
29     if(l1==null) return l2;
30     if(l2==null) return l1;
```

Saved | Ln 38, Col 2

Testcase | **Test Result**

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

`lists =`
`[[1,4,5],[1,3,4],[2,6]]`

Output

`[1,1,2,3,4,4,5,6]`

Executed