

Assignment – 3

1. PrintLinked list

Code:

```
class Solution {  
    void printList(Node head) {  
  
        Node temp = head;  
        while(temp != null){  
            System.out.print(temp.data+" ");  
            temp = temp.next;  
        }  
    }  
}
```

The screenshot displays a coding platform interface. On the left, the 'Output Window' shows 'Compilation Results' for 'Custom Input' by 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Below this, statistics are shown: 'Test Cases Passed: 1112 / 1112', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 1 / 1', and 'Time Taken: 1.74'. A 'Solve Next' button is at the bottom left. On the right, the code editor shows the solution code for the 'PrintLinked list' problem, which matches the code provided in the previous block. The code is for a class 'Solution' with a method 'printList' that traverses a linked list and prints its elements.

2. Remove duplicates from sorted list

Code:

```
class Solution {  
    public ListNode deleteDuplicates(ListNode head) {  
        ListNode temp=head;  
        while(temp!=null&&temp.next!=null){
```

```

        if(temp.val==temp.next.val){
            temp.next=temp.next.next;
        }
        else{
            temp=temp.next;
        }
    }
    return head;
}
}

```

Description
Editorial
Solutions

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*

9.1K 106 78 Online

Solved

19

20

21

22

23

24

25

temp=temp.next;
}
}
return head;
}

Ln 21, Col 10 | Saved

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Input

head =
[1,1,2]

3. Reverse a LinkedList

Code:

```

class Solution {

    public ListNode reverseList(ListNode head) {

        ListNode prev = null;

        ListNode temp = head;

        while (temp != null) {

```

```

        ListNode nextNode = temp.next;

        temp.next = prev;

        prev = temp;

        temp = nextNode;

    }

    return prev;

}

```

The screenshot shows the LeetCode interface for problem 206, "Reverse Linked List". The problem is marked as "Solved" with a green checkmark. The difficulty is "Easy". The description states: "Given the head of a singly linked list, reverse the list, and return the reversed list." The input for the test case is "head = [1,2,3,4,5]". The solution is written in Java and is marked as "Accepted" with a runtime of 0 ms. The code in the editor is as follows:

```

18     prev = temp;
19     temp = nextNode;
20 }
21 return prev;
22 }
23
24 }

```

4. Delete Middle node of a list

Code:

```

class Solution {

    public ListNode deleteMiddle(ListNode head) {

        if(head.next==null){

            return null;

```

```

    }

    ListNode first=head.next;

    ListNode second=head;

    while(first.next!=null&& first.next.next!=null){

        first=first.next.next;

        second =second.next;

    }

    second.next=second.next.next;

    return head;

}
}

```

Description
Editorial
Solutions

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*

4.4K 71 48 Online

Testcase
Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```

19 while(first.next!=null&& first.n
20     first=first.next.next;
21     second =second.next;
22 }
23 second.next=second.next.next;
24 return head;
25
26
27
28 }

```

Ln 26, Col 1 | Saved

5. Merge two sorted linked lists

Code:

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

        if(list1!=null && list2!=null){
            if(list1.val<list2.val){
                list1.next=mergeTwoLists(list1.next,list2);
                return list1;
            }
            else{
                list2.next=mergeTwoLists(list1,list2.next);
                return list2;
            }
        }
        if(list1==null)
            return list2;
        return list1;
    }
}
```

The screenshot shows the LeetCode interface for the problem '21. Merge Two Sorted Lists'. The problem is marked as 'Easy' and 'Solved'. The description states: '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.' The interface includes tabs for 'Description', 'Editorial', and 'Solutions'. On the right, a code editor shows a Java solution. At the bottom, the status 'Accepted' is displayed with a runtime of 0 ms, and three test cases are listed: 'Case 1', 'Case 2', and 'Case 3'.

21. Merge Two Sorted Lists

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.

22.9K 411 445 Online

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

6. Remove duplicates from sorted lists 2

Code:

```
class Solution {  
    public ListNode deleteDuplicates(ListNode head) {  
        if (head == null || head.next == null) return head;  
        ListNode res = new ListNode(0, head);  
        ListNode prev = res;  
        while(head != null && head.next != null){  
            if(head.next.val == head.val){  
                while(head.next != null && head.next.val == head.val){  
                    head = head.next;  
                }  
                prev.next = head.next;  
            }else{  
                prev = prev.next;  
            }  
            head = head.next;  
        }  
        return res.next;  
    }  
}
```

Description
Editorial
Solutions

82. Remove Duplicates from Sorted List II

Solved

Medium Topics Companies

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.

9.1K 82 72 Online

Ln 29, Col 13 | Saved

Testcase
Test Result

```

21         head = head.next;
22     }
23     prev.next = head.next;
24 }else{
25     prev = prev.next;
26 }
27     head = head.next;
28 }
29     return res.next;
30 }
31 }

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

7. Detect a cycle in a linked list

Code:

```

public class Solution {

    public boolean hasCycle(ListNode head) {

        if(head==null || head.next==null){

            return false;

        }

        ListNode temp=head;

        ListNode prev=head;

        while(temp!=null&&temp.next!=null){

            temp=temp.next.next;

            prev=prev.next;

            if(temp==prev){

```

```

        return true;
    }

}

return false;
}
}

```

Description
Editorial
Solutions

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 `next` pointer of the last node points to. `pos = -1` means the list has no cycle.

16.2K 352 197 Online

Solved

Testcase
Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

```

11         temp=temp.next.next;
12         prev=prev.next;
13         if(temp==prev){
14             return true;
15         }
16     }
17
18     return false;
19 }
20
21 }

```

Ln 15, Col 14 | Saved

8. Reverse linked list 2

Code:

```

class Solution {

    public ListNode reverseBetween(ListNode head, int left, int right) {

        ListNode dummy = new ListNode(0);

        dummy.next = head;

        ListNode prev = dummy;

        for(int i = 0; i < left - 1; i++)

```



```

        prev = prev.next;

        ListNode curr = prev.next;

        for(int i = 0; i < right - left; i++){

            ListNode forw = curr.next;

            curr.next = forw.next;

            forw.next = prev.next;

            prev.next = forw;

        }

        return dummy.next;

    }
}

```

Description
Editorial
Solutions

92. Reverse Linked List II

Medium

Given the `head` of a singly linked list and two integers `left` and `right` where `left <= right`, reverse the nodes of the list between `left` and `right`, inclusive, and return the list.

12K 151 102 Online

Solved

Testcase
Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

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

```

1 class Solution {
2     public ListNode reverseBetween(ListNode head, int left, int right) {
3         ListNode dummy = new ListNode(0);
4         dummy.next = head;
5         ListNode prev = dummy;
6         for(int i = 0; i < left - 1; i++)
7             prev = prev.next;
8         // reverse the list between left and right
9     }
10 }

```

9. rotate a list

Code:

```

class Solution {

    public ListNode rotateRight(ListNode head, int k) {

```

```
    if (head==null){  
        return head;  
    }  
    int length=1;  
    ListNode tail=head;  
    ListNode cur=head;  
    while (tail.next!=null){  
        tail=tail.next;  
        length++;  
    }  
    k=k%length;  
    if (k==0){  
        return head;  
    }  
    for(int i=0;i<length-k-1;i++){  
        cur=cur.next;  
    }  
    ListNode newh=cur.next;  
    cur.next=null;  
    tail.next=head;  
    return newh;  
}  
}
```

Description
Editorial
Solutions

61. Rotate List

Medium Topics Companies

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

10.2K 101 68 Online

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

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

Java Auto

```

29     }
30     ListNode newh=cur.next;
31     cur.next=null;
32     tail.next=head;
33     return newh;
34 }
35 }

```

Ln 31, Col 21 | Saved

Run

10. Sort List

Code:

```

class Solution {

    public ListNode sortList(ListNode head) {

        if(head==null || head.next==null){

            return head;

        }

        ListNode middle= middleNode(head);

        ListNode start2= middle.next;

        middle.next=null;

        ListNode first = sortList(head);

        ListNode second = sortList(start2);

```

```

return mergeTwoLists(first,second);

}

public ListNode middleNode(ListNode head) {

    ListNode slow=head;

    ListNode fast=head;

    while(fast!=null &&fast.next!=null && fast.next.next!=null){

        slow=slow.next;

        fast=fast.next.next;

    }

    return slow;

}

public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

    ListNode list3 = new ListNode();

    ListNode temp3 = list3;

    while (list1 != null && list2 != null) {

        if (list1.val <= list2.val) {

            temp3.next = list1;

            list1 = list1.next;

        } else {

            temp3.next = list2;

            list2 = list2.next;

        }

    }

```

```

        temp3 = temp3.next;
    }
    if (list1 != null) {
        temp3.next = list1;
    } else {
        temp3.next = list2;
    }
    return list3.next;
}
}

```

148. Sort List

Medium Topics Companies

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

Format: 4.

12.1K 108 ☆ ↗ ?

Solved ✓

89 Online

Ln 61, Col 2 | Saved

🔥

Run

Submit

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

```

31         slow=slow.next;
32         fast=fast.next.next;
33     }
34     return slow;
35 }
36
37 public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
38     ListNode list3 = new ListNode();
39     ListNode temp3 = list3;
40

```

11. Detect a cycle in a linked list 2

Code:

```

public class Solution {
    public ListNode detectCycle(ListNode head) {

```

```

ListNode slow = head, fast = head;

while (fast != null && fast.next != null) {

    slow = slow.next;

    fast = fast.next.next;

    if (slow == fast) break;

}

if (fast == null || fast.next == null) return null;

while (head != slow) {

    head = head.next;

    slow = slow.next;

}

return head;

}

}

```

Description
Editorial
Solutions

142. Linked List Cycle II

Solved

Medium Topics Companies

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

14.1K 179 102 Online

Ln 20, Col 34 | Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```

17         fast = fast.next.next;
18         if (slow == fast) break;
19     }
20     if (fast == null || fast.next == null)
21     while (head != slow) {
22         head = head.next;
23         slow = slow.next;
24     }
25     return head;
26 }
27 }

```