WORKSHEET 2

Student Name: Prashant Kumar | UID: 22BCS15537

Branch: BE-CSE | Section/Group: 22BCS_NTPP-602-A

Semester: 6 th | Date of Performance: 22/01/2025

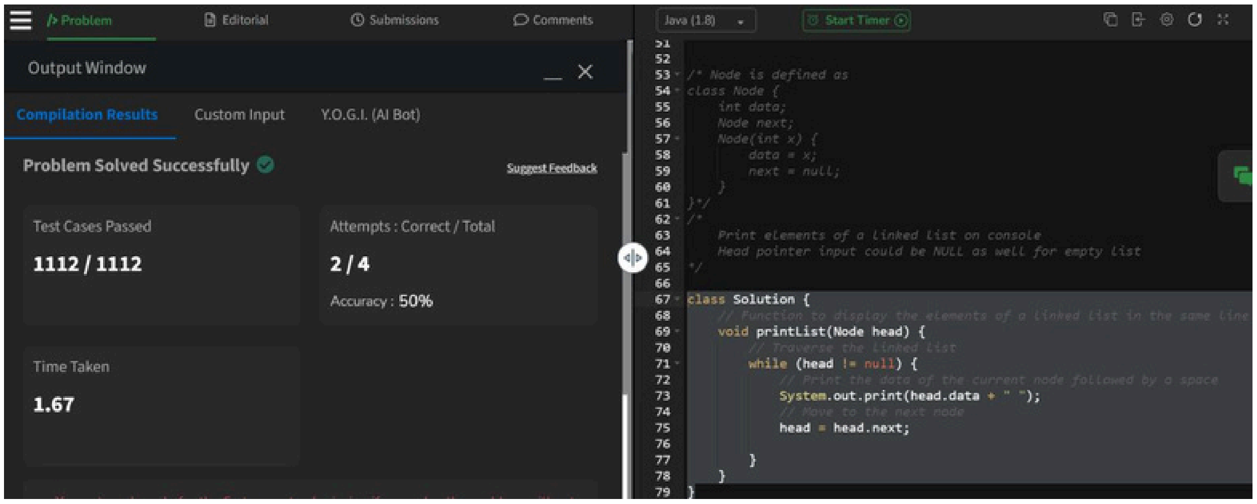
Subject Name: AP LAB - II | Subject Code: 22CSP-351

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

2. Source Code:

```
class Solution {

   // Function to display the elements of a linked list in the same line
   void printList(Node head) {
      // Traverse the linked list
      while (head != null) {
         // Print the data of the current node followed by a space
         System.out.print(head.data + " ");
         // Move to the next node
         head = head.next;

      }   }}
```

3. Screenshots of outputs:



Problem Solved Successfully

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
2 / 4
Accuracy : 50%

Time Taken
1.67

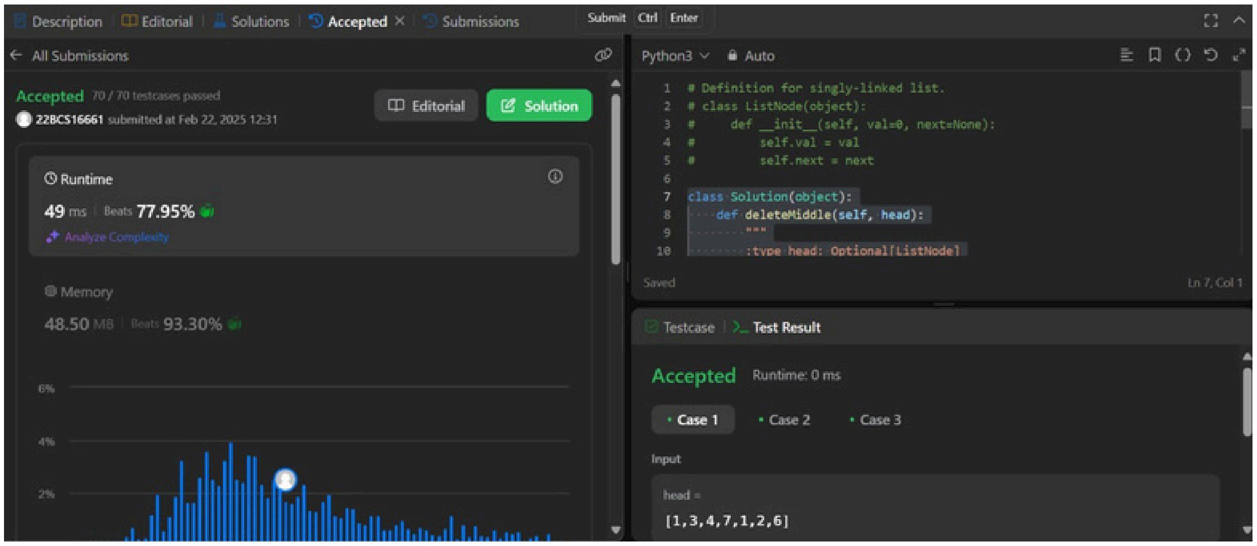
```
/* Node is defined as
class Node {
    int data;
    Node next;
    Node(int x) {
        data = x;
        next = null;
    }
}*/
/*
    Print elements of a linked list on console
    Head pointer input could be NULL as well for empty list
*/

class Solution {
    // Function to display the elements of a linked list in the same line
    void printList(Node head) {
        // Traverse the linked list
        while (head != null) {
            // Print the data of the current node followed by a space
            System.out.print(head.data + " ");
            // Move to the next node
            head = head.next;

        }
    }
}
```

2.
Aim: You are given the head of a linked list. Delete the middle node, and return *the* head *of the modified linked list*

Source Code:

```
class Solution(object):
    def deleteMiddle(self, head):
        """
        :type head: Optional[ListNode]
        :rtype: Optional[ListNode]
        """
       if not head or not head.next:
           return None
        slow = head
        fast = head
        prev = None
        while fast and fast.next:
        fast = fast.next.next
        prev = slow
        slow = slow.next
        prev.next = slow.next
        return head
```

Screenshots of outputs:



Accepted 70 / 70 testcases passed
22BCS16661 submitted at Feb 22, 2025 12:31

Runtime
49 ms | Beats 77.95%

Memory
48.50 MB | Beats 93.30%

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next

class Solution(object):
    def deleteMiddle(self, head):
        """
        :type head: Optional[ListNode]
```

Saved Ln 7, Col 1

Accepted Runtime: 0 ms

Input

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

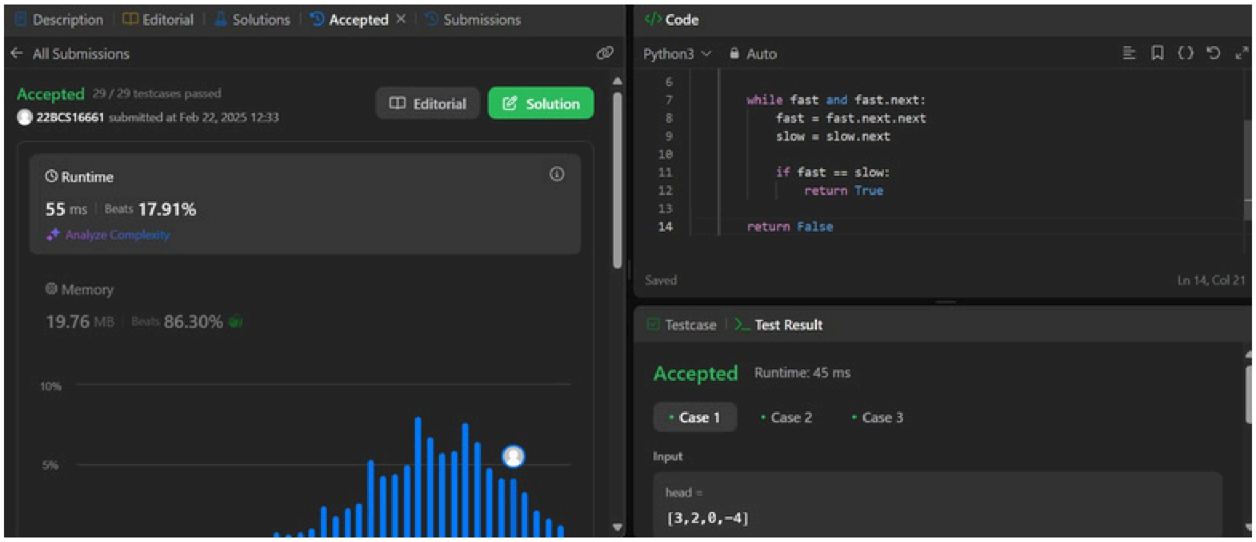
3.

Aim: 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.

Source Code:

```
class Solution:

    def hasCycle(self, head: Optional[ListNode]) -> bool:

        fast = head
        slow = head
        while fast and fast.next:

            fast = fast.next.next
            slow = slow.next
            if fast == slow:

                return True

        return False
```

4. Screenshots of outputs:



Accepted 29 / 29 testcases passed
22BCS16661 submitted at Feb 22, 2025 12:33

Runtime
55 ms | Beats 17.91%

Memory
19.76 MB | Beats 86.30%

```
    while fast and fast.next:
        fast = fast.next.next
        slow = slow.next

        if fast == slow:
            return True

    return False
```

Accepted Runtime: 45 ms

Input

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