Find middle element in a Linked List

Problem Statement: Given the **head** of a singly linked list, return *the middle node of the linked list*. If there are two middle nodes, return the second middle node.

Examples:

Input Format:

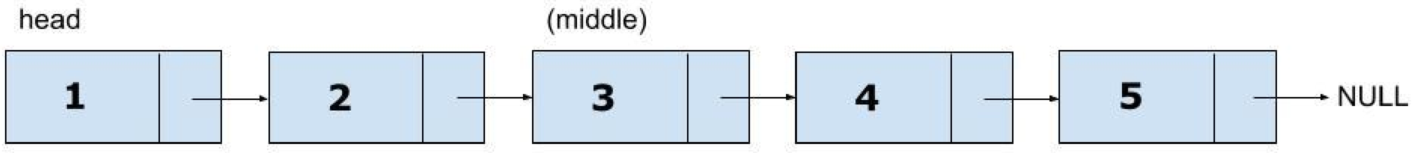
```
( Pointer / Access to the head of a Linked list )

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

Result: `[3,4,5]`

```
( As we will return the middle of Linked list the further linked list will be still
available )
```

Explanation: `The middle node of the list is node 3 as in the below image.`



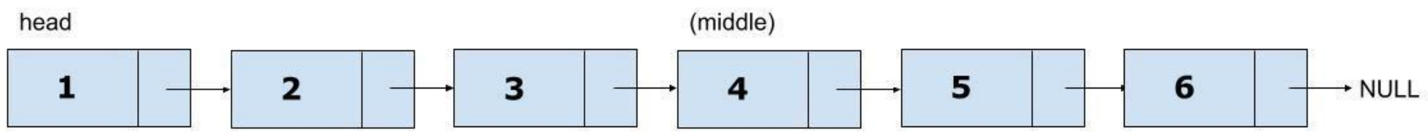
Input Format:

```
Input: head = [1,2,3,4,5,6]
```

Result: `[4,5,6]`

Explanation:

```
Since the list has two middle nodes with values 3 and 4, we return the second one.
```



Solution

Disclaimer: Don't jump directly to the solution, try it out yourself first.

Solution 1: Naive Approach

Intuition: We can traverse through the Linked List while maintaining a count of nodes let's say in variable **n** , and then traversing for 2nd time for **n/2** nodes to get to the middle of the list.

Source Code:

```
ListNode middleNode(ListNode head) {
        int n = 0;
        ListNode temp = head;
        while(temp) {
        n++;
                temp = temp.next;
        }
        temp = head;
        for(int i = 0; i < n / 2; i++) {
                temp = temp.next;
        }
        return temp;
        }
```

Solution 2: [Efficient] Tortoise-Hare-Approach

Unlike the above approach, we don't have to maintain nodes count here and we will be able to find the middle node in a single traversal so this approach is more efficient.

Intuition: In the Tortoise-Hare approach, we increment slow ptr by 1 and fast ptr by 2, so if take a close look fast ptr will travel double than that of the slow pointer. So when the fast ptr will be at the end of Linked List, slow ptr would have covered half of Linked List till then. So slow ptr will be pointing towards the middle of Linked List.

Approach:

1. Create two pointers slow and fast and initialize them to a head pointer.
2. Move slow ptr by one step and simultaneously fast ptr by two steps until fast ptr is NULL or next of fast ptr is NULL.
3. When the above condition is met, we can see that the slow ptr is pointing towards the middle of Linked List and hence we can return the slow pointer.

Source Code:

```
public ListNode middleNode(ListNode head) {
        ListNode fast=head,slow=head;
        while(fast!=null&&fast.next!=null)
        {
            slow=slow.next;
            fast=fast.next.next;
        }
        return slow;
    }
```