Find middle element of a LinkedList :-

Approach 1 :

```
public class FindMiddleOfLinkedList {
	public static Node findMiddle(Node head) {
		if(head == null) {
			return null;
		}
		if(head.next==null) {
			return head;
		}
		Node p = head;
		int lenght = LinkedListLength.getLinkedListLength(head);
		int middle = (lenght/2)-1;
		int count = 0;
		while(count<middle) {
			p=p.next;
			count++;
		}
		return p;
	}
	public static void main(String[] args) {
		Node head = new Node(28);
		Node node27 = new Node(27);
		Node node26 = new Node(26);
		Node node25 = new Node(25);
		Node node24 = new Node(24);
		head.next = node27;
		node27.next = node26;
		node26.next = node25;
		node25.next = node24;
		Node middleNode = findMiddle(head);
		System.out.println(middleNode.data);
	}
}
```

```
Time Complexity -> O(n)+O(n/2) =O(n)

Space Complexity -> O(1)
```

Approach 2:-

```
	private static Node getMiddleElementOfLinkedList(Node head) {
		if (head == null)// Base condition
			return null;
		Node p = head;// slow pointer
		Node q = head;// fast pointer
		while (q != null && q.next != null) {
			p = p.next;
			q = q.next.next;
		}
		return p;
	}
```

Time Complexity -> O(n/2) =O(n)

Space Complexity -> O(1)

Is Circular Linked List

```
public class IsCircularLinkedList {

	public static boolean isCircular(Node head) {
		if (head == null || head.next == null || head.next.next == null) {
			return false;
		}
		Node p = head;// slow pointer
		Node q = head;// fast pointer
		while (q != null && q.next != null) {
			p = p.next;
			q = q.next.next;
			if (p == q) {
				return true;
			}
		}
		return false;
	}


	public static void main(String[] args) {
		Node head = new Node(28);
		Node node28 = head;
		Node node27 = new Node(27);
		Node node26 = new Node(26);
		Node node25 = new Node(25);
		Node node24 = new Node(24);
		head.next = node27;
		node27.next = node26;
		node26.next = node25;
		node25.next = node24;
		node24.next = node28;
		boolean isCircular = isCircular(head);
		System.out.println(isCircular);
	}
}
```

Find if Loop exists: -

```
    static boolean hasCycle(SinglyLinkedListNode head) {
        if (head == null || head.next == null || head.next.next == null) {
            return false;
        }
        SinglyLinkedListNode p = head;// slow pointer
        SinglyLinkedListNode q = head;// fast pointer
        while (q != null && q.next != null) {
            p = p.next;
            q = q.next.next;
            if (p == q) {
                return true;
            }
        }
        return false;
    }
```

Link:

https://www.hackerrank.com/challenges/detect-whether-a-linked-list-contains-a-cycle/problem