14TH AUGUST LINKEDLIST

Q1. Check if a key is present in a linked list:

```
java
Copy code
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList {
    Node head;

    public boolean search(Node head, int key) {
        Node current = head;
        while (current != null) {
            if (current.data == key)
                return true;
            current = current.next;
        }
        return false;
    }
}
```

Q2. Insert a node at a given position in a linked list:

```
java
Copy code
class Node {
    int data;
    Node next;
```

```
    Node(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList {
    Node head;

    public void insertAfter(Node prevNode, int newData) {
        if (prevNode == null) {
            System.out.println("Previous node cannot be null.");
            return;
        }
        Node newNode = new Node(newData);
        newNode.next = prevNode.next;
        prevNode.next = newNode;
    }
}
```

Q3. Remove duplicates from a sorted linked list:

```
java
Copy code
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList {
    Node head;
```

```
    public Node deleteDuplicates(Node head) {
        Node current = head;
        while (current != null && current.next != null) {
            if (current.data == current.next.data) {
                current.next = current.next.next;
            } else {
                current = current.next;
            }
        }
        return head;
    }
}
```

Q4. Check if a linked list is a palindrome:

```
java
Copy code
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList {
    Node head;

    public boolean isPalindrome(Node head) {
        Node slow = head, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
```

```
        Node secondHalf = reverse(slow);
        Node firstHalf = head;

        while (secondHalf != null) {
            if (firstHalf.data != secondHalf.data)
                return false;
            firstHalf = firstHalf.next;
            secondHalf = secondHalf.next;
        }

        return true;
    }

    private Node reverse(Node head) {
        Node prev = null;
        while (head != null) {
            Node nextNode = head.next;
            head.next = prev;
            prev = head;
            head = nextNode;
        }
        return prev;
    }
}
```

Q5. Sum of two numbers represented by linked lists:

```
java
Copy code
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        next = null;
    }
```

```
}

class LinkedList {
    Node head;

    public Node addTwoNumbers(Node l1, Node l2) {
        Node dummy = new Node(0);
        Node current = dummy;
        int carry = 0;

        while (l1 != null || l2 != null) {
            int sum = carry;
            if (l1 != null) {
                sum += l1.data;
                l1 = l1.next;
            }
            if (l2 != null) {
                sum += l2.data;
                l2 = l2.next;
            }

            carry = sum / 10;
            current.next = new Node(sum % 10);
            current = current.next;
        }

        if (carry > 0) {
            current.next = new Node(carry);
        }

        return dummy.next;
    }
}
```