Linked List - 4

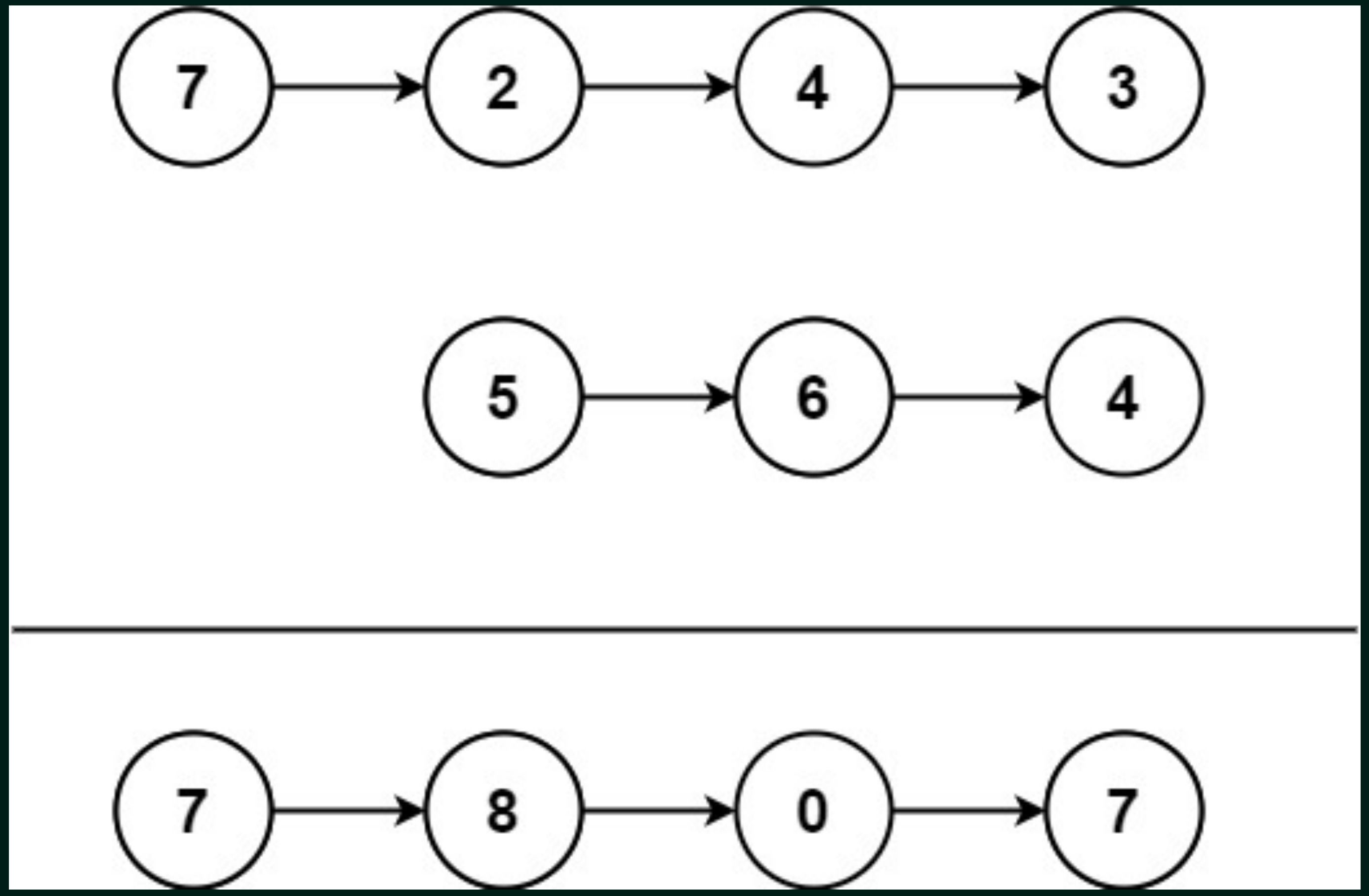
In This Lecture

1. Add Two Numbers as a List
2. Reorder List

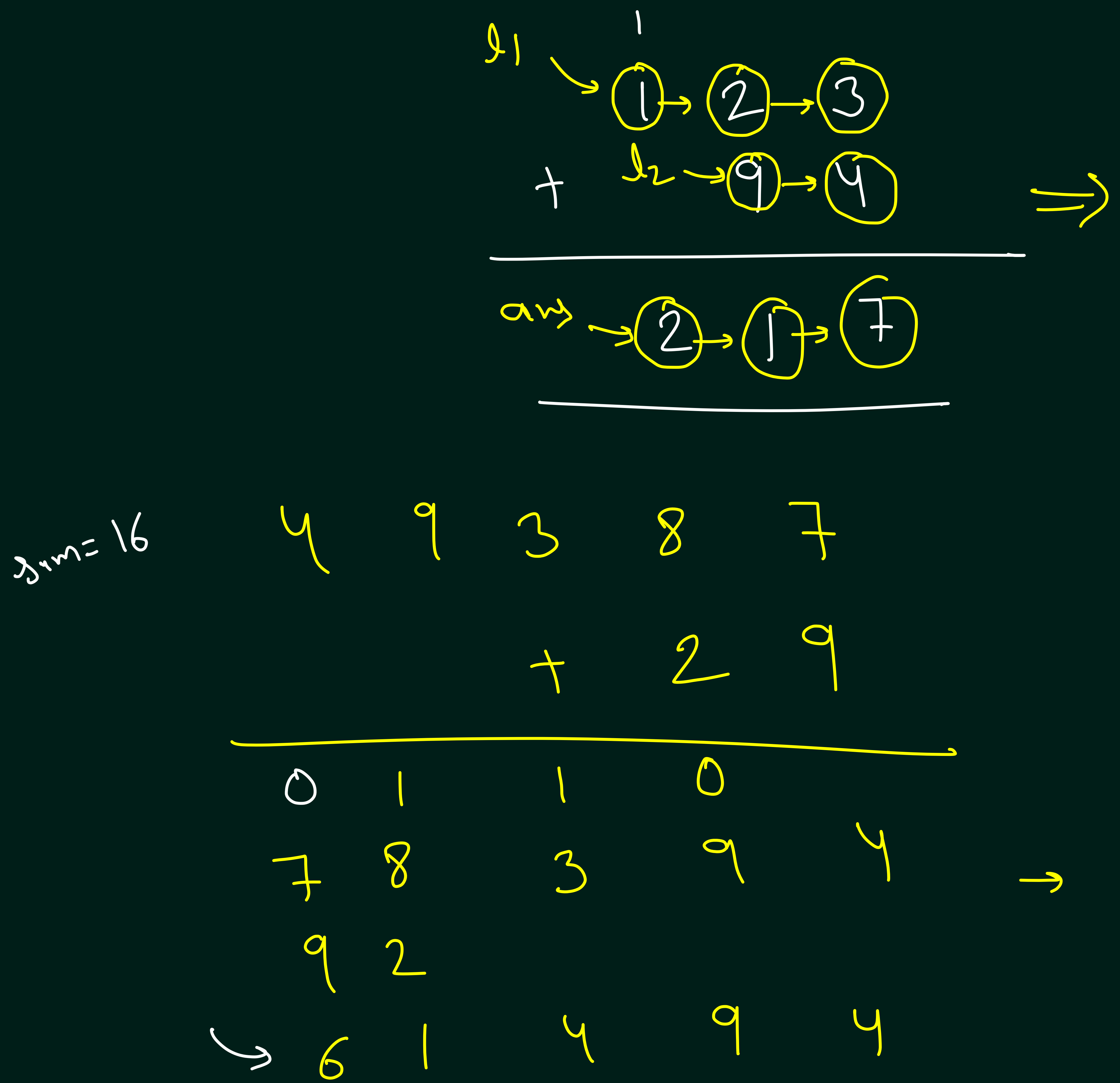
Add Two Numbers

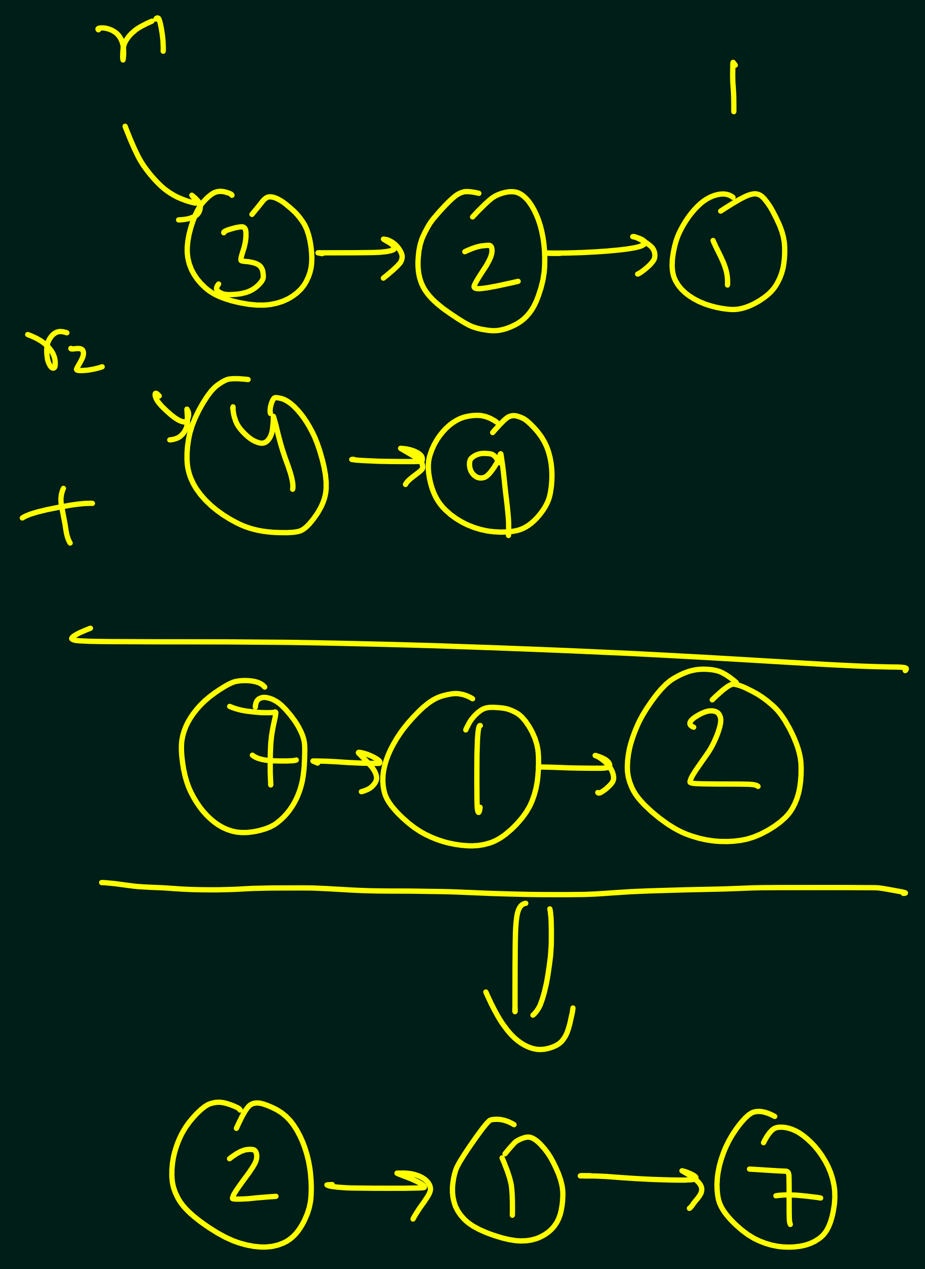
You are given two non-empty linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

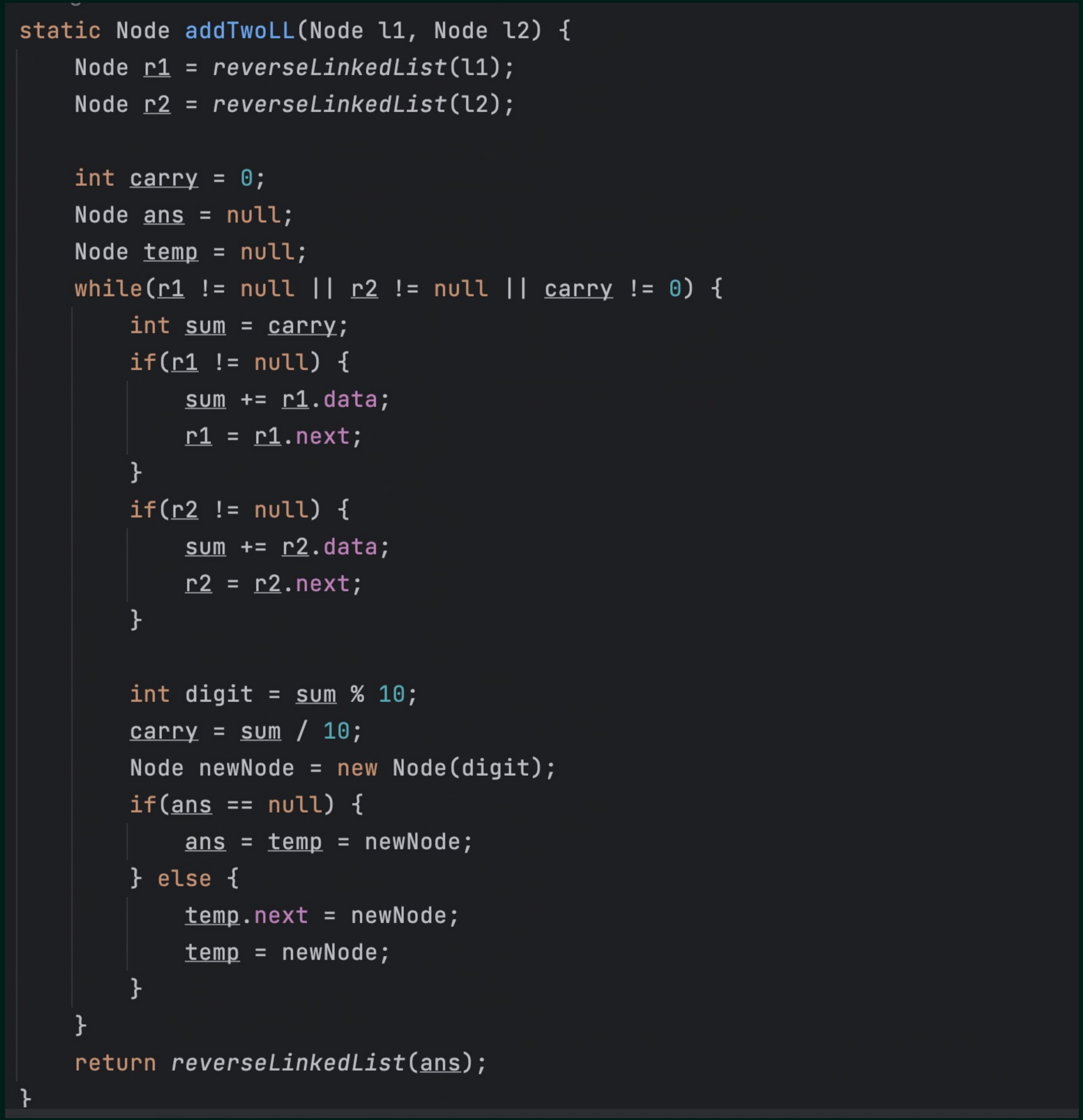


Add Two Numbers





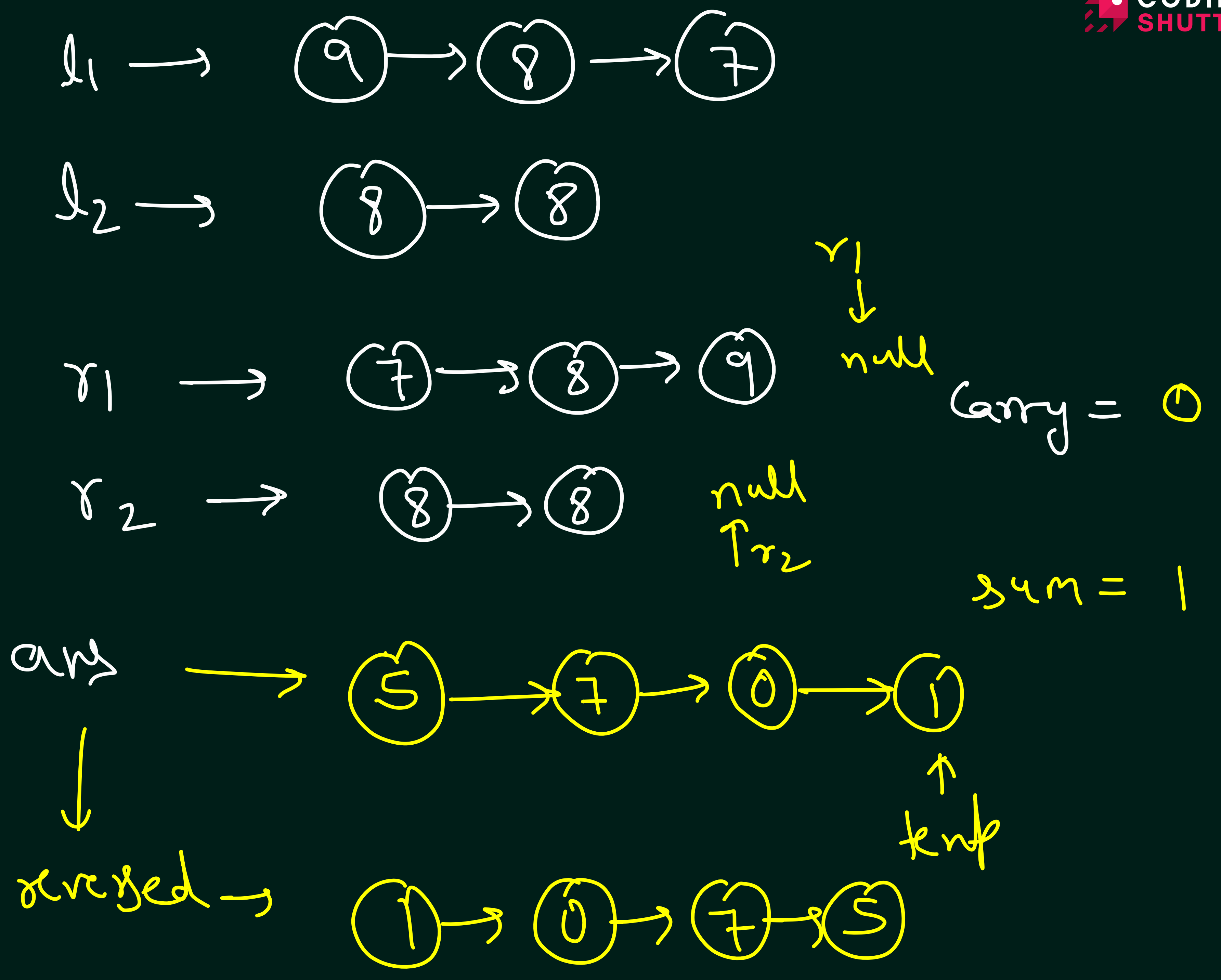
Add Two Numbers



```java
static Node addTwoLL(Node l1, Node l2) {
    Node r1 = reverseLinkedList(l1);
    Node r2 = reverseLinkedList(l2);

    int carry = 0;
    Node ans = null;
    Node temp = null;
    while(r1 != null || r2 != null || carry != 0) {
        int sum = carry;
        if(r1 != null) {
            sum += r1.data;
            r1 = r1.next;
        }
        if(r2 != null) {
            sum += r2.data;
            r2 = r2.next;
        }

        int digit = sum % 10;
        carry = sum / 10;
        Node newNode = new Node(digit);
        if(ans == null) {
            ans = temp = newNode;
        } else {
            temp.next = newNode;
            temp = newNode;
        }
    }
    return reverseLinkedList(ans);
}
```



Reorder List

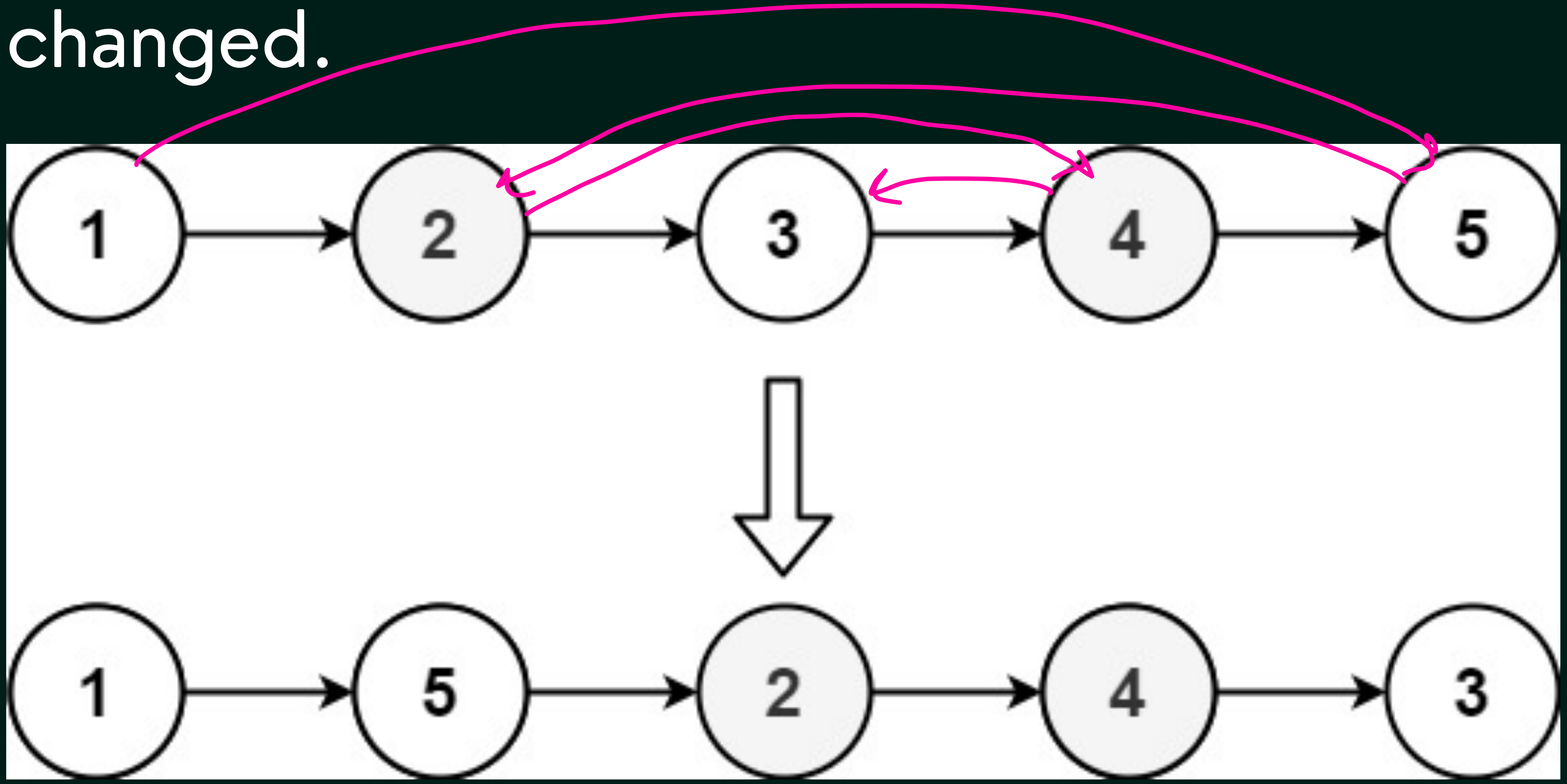
You are given the head of a singly linked-list. The list can be represented as:

✓ L0 → L1 → ... → Ln - 1 → Ln

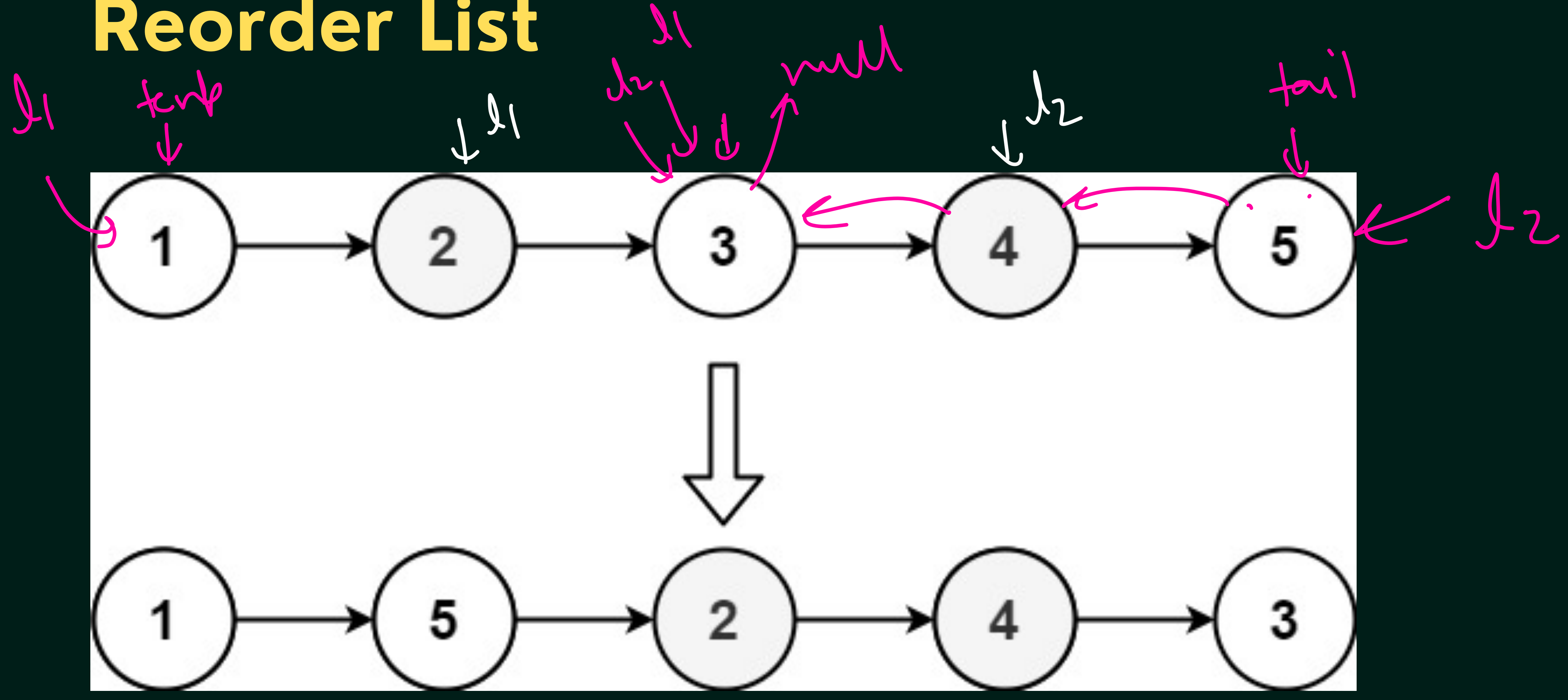
Reorder the list to be on the following form:

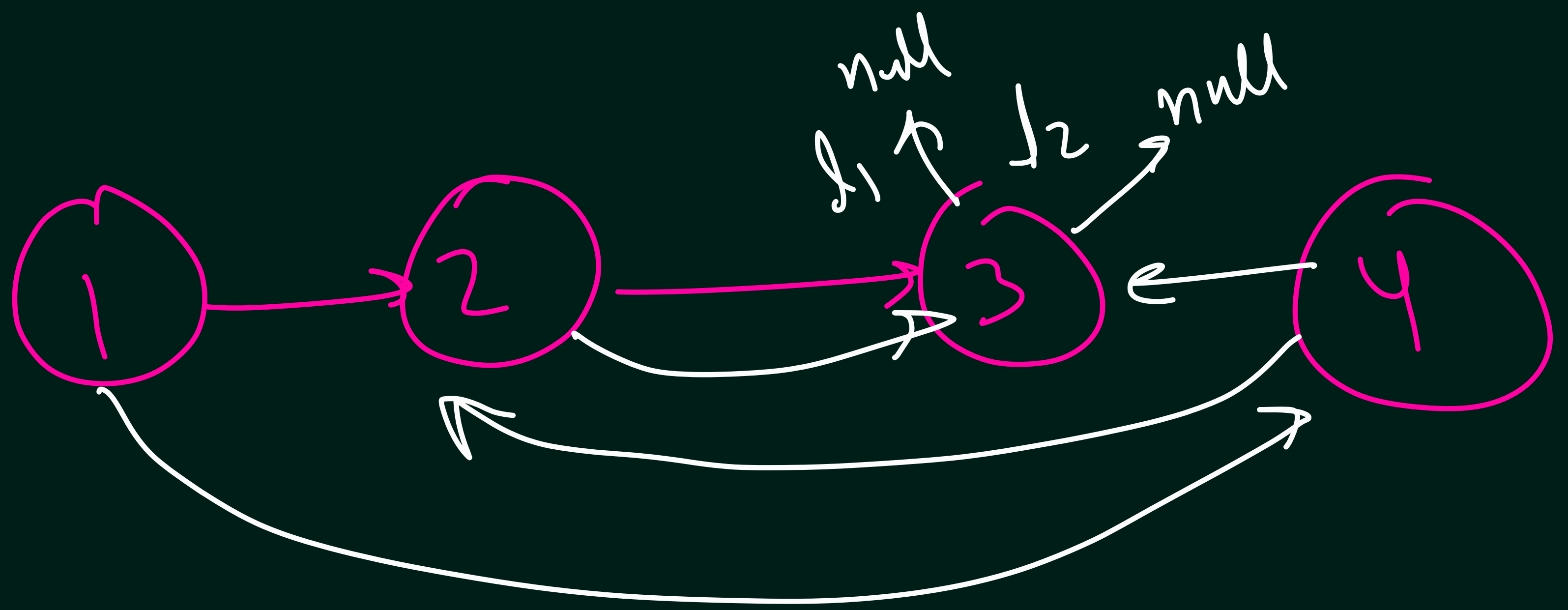
L0 → Ln → L1 → Ln - 1 → L2 → Ln - 2 → ...

You may not modify the values in the list's nodes. Only nodes themselves may be changed.



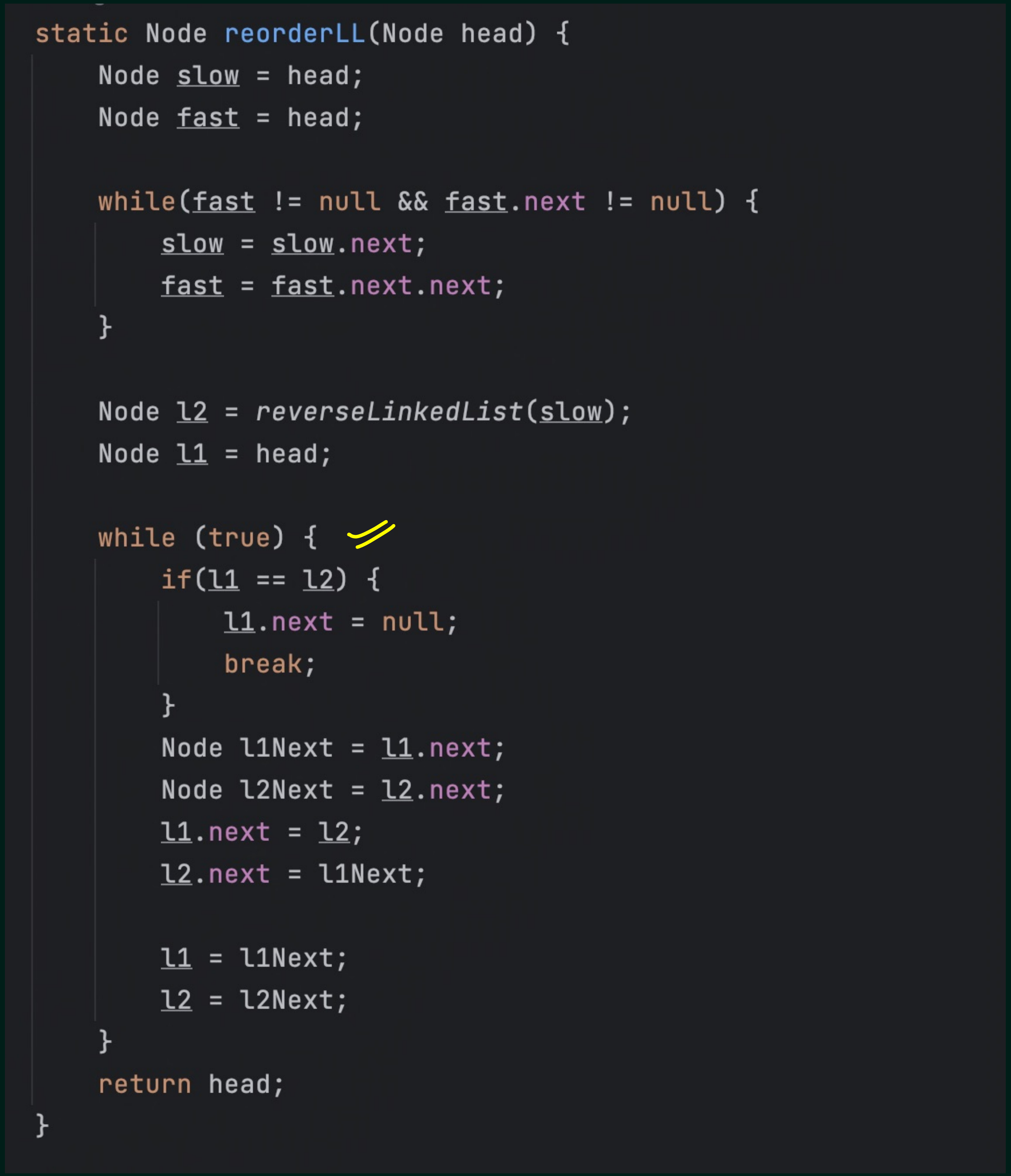
Reorder List





```
{
    l1.next = l2
    l2.next = l1.next
    l1 = l1.next
    l2 = l2.next
}
```

Reorder List



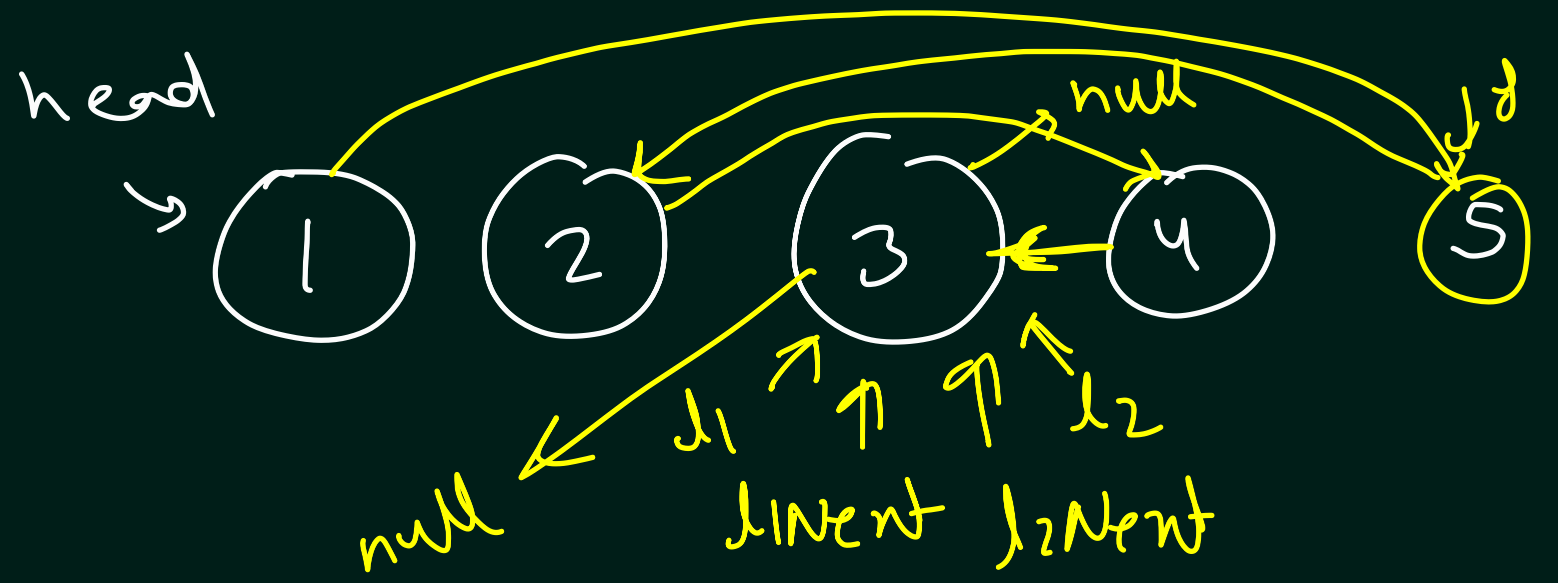
```java
static Node reorderLL(Node head) {
    Node slow = head;
    Node fast = head;

    while(fast != null && fast.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }

    Node l2 = reverseLinkedList(slow);
    Node l1 = head;

    while (true) {
        if(l1 == l2) {
            l1.next = null;
            break;
        }
        Node l1Next = l1.next;
        Node l2Next = l2.next;
        l1.next = l2;
        l2.next = l1Next;

        l1 = l1Next;
        l2 = l2Next;
    }
    return head;
}
```


head
null
1
2
3
4
5
null
l1
l2
l1Next
l2Next