(Linked List)

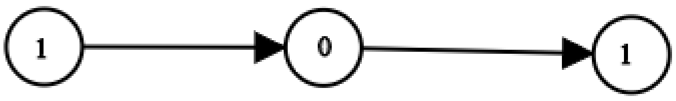
Problem-1

Given `head` which is a reference node to a singly-linked list. The value of each node in the linked list is either `0` or `1`. The linked list holds the binary representation of a number.

Return the *decimal value* of the number in the linked list.

The **most significant bit** is at the head of the linked list.

Example 1:



```
Input: head = [1,0,1]
Output: 5
Explanation: (101) in base 2 = (5) in base 10
```

Example 2:

```
Input: head = [0]
Output: 0
```

Constraints:

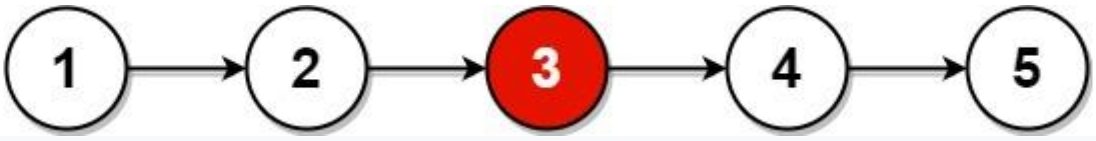
- The Linked List is not empty.
- Number of nodes will not exceed `30`.
- Each node's value is either `0` or `1`.

Problem-2

Given the `head` of a singly linked list, return *the middle node of the linked list*.

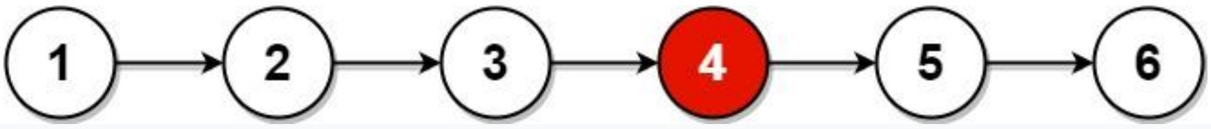
If there are two middle nodes, return **the second middle** node.

Example 1:



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

Output: [3,4,5]

Explanation: The middle node of the list is node 3.
```

Example 2:



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

Output: [4,5,6]

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

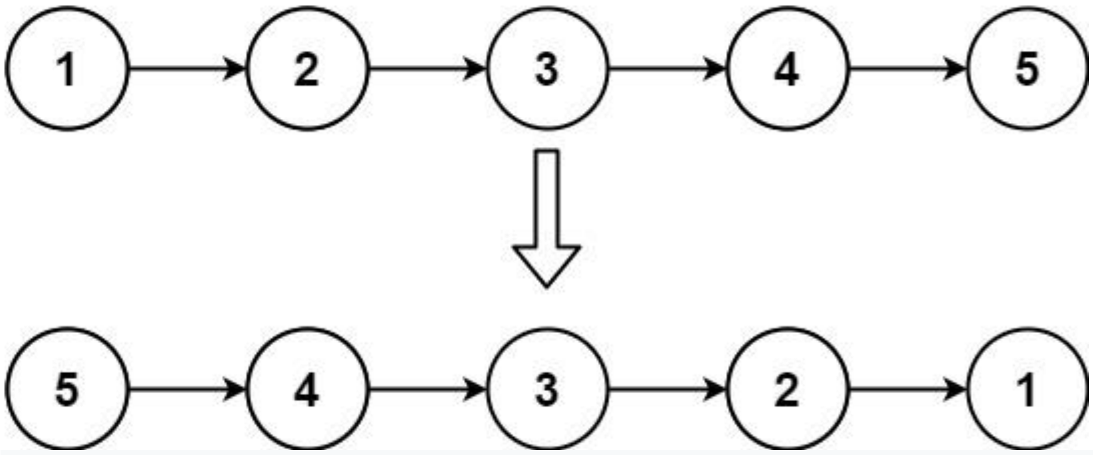
Constraints:

- The number of nodes in the list is in the range `[1, 100]`.
- `1 <= Node.val <= 100`

Problem-3

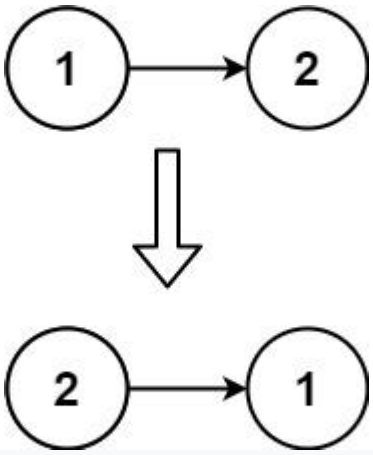
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:



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

Output: [5,4,3,2,1]
```

Example 2:



```
Input: head = [1,2]

Output: [2,1]
```

Example 3:

```
Input: head = []

Output: []
```

Constraints:

- The number of nodes in the list is the range `[0, 5000]`.
- `-5000 <= Node.val <= 5000`

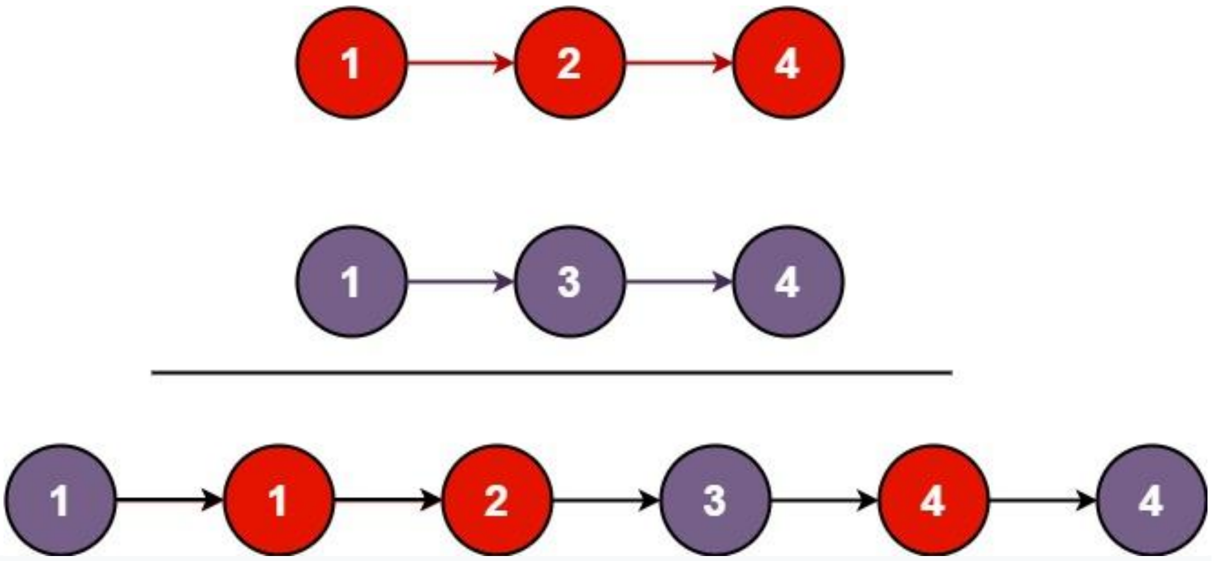
Problem-4

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists in a one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

Example 1:



```
Input: list1 = [1,2,4], list2 = [1,3,4]

Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []
```

```
Output: []
```

Example 3:

```
Input: list1 = [], list2 = [0]

Output: [0]
```

Constraints:

- The number of nodes in both lists is in the range `[0, 50]`.
- `-100 <= Node.val <= 100`
- Both `list1` and `list2` are sorted in **non-decreasing** order.

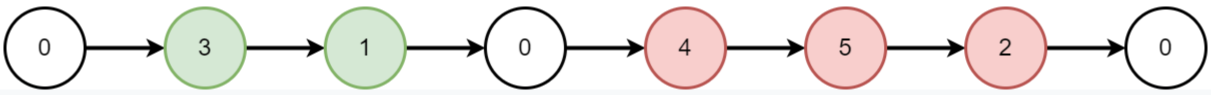
Problem-5

You are given the `head` of a linked list, which contains a series of integers **separated** by `0`'s. The **beginning** and **end** of the linked list will have `Node.val == 0`.

For **every** two consecutive `0`'s, **merge** all the nodes lying in between them into a single node whose value is the **sum** of all the merged nodes. The modified list should not contain any `0`'s.

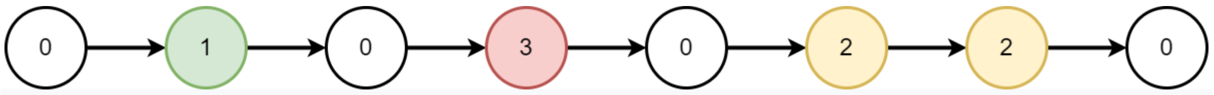
Return *the* `head` *of the modified linked list.*

Example 1:



```
Input: head = [0,3,1,0,4,5,2,0]

Output: [4,11]

Explanation:

The above figure represents the given linked list. The modified list contains

- The sum of the nodes marked in green: 3 + 1 = 4.

- The sum of the nodes marked in red: 4 + 5 + 2 = 11.
```

Example 2:



```
Input: head = [0,1,0,3,0,2,2,0]

Output: [1,3,4]

Explanation:

The above figure represents the given linked list. The modified list contains

- The sum of the nodes marked in green: 1 = 1.

- The sum of the nodes marked in red: 3 = 3.

- The sum of the nodes marked in yellow: 2 + 2 = 4.
```

Constraints:

- The number of nodes in the list is in the range `[3, 2 * 10⁵]`.
- `0 <= Node.val <= 1000`
- There are **no** two consecutive nodes with `Node.val == 0`.
- The **beginning** and **end** of the linked list have `Node.val == 0`.