Linked List-2

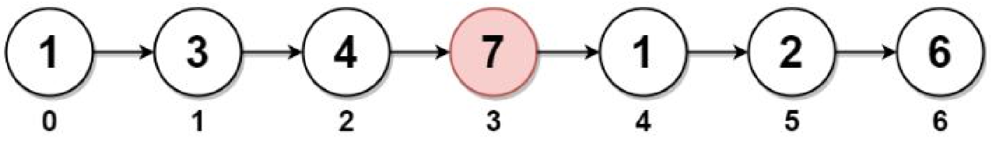
Assignment Solution

SKILLS

1. **You are given the head of a linked list. Delete the middle node, and return the head of the modified linked list. [Leetcode 2095]**

The middle node of a linked list of size n is the ⌊n / 2⌋th node from the start using 0-based indexing, where ⌊x⌋ denotes the largest integer less than or equal to x .
For n = 1 , 2 , 3 , 4 , and 5 , the middle nodes are 0 , 1 , 1 , 2 , and 2 , respectively.

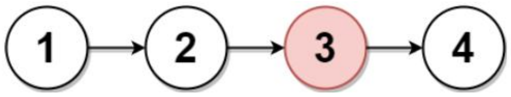
Example 1:



Input: head = [1,3,4,7,1,2,6]
Output: [1,3,4,1,2,6]

Explanation:
The above figure represents the given linked list. The indices of the nodes are written below.
Since n = 7, node 3 with value 7 is the middle node, which is marked in red.
We return the new list after removing this node.

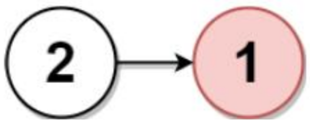
Example 2:



Input: head = [1,2,3,4]
Output: [1,2,4]

Explanation:
The above figure represents the given linked list.
For n = 4, node 2 with value 3 is the middle node, which is marked in red.

Example 3:



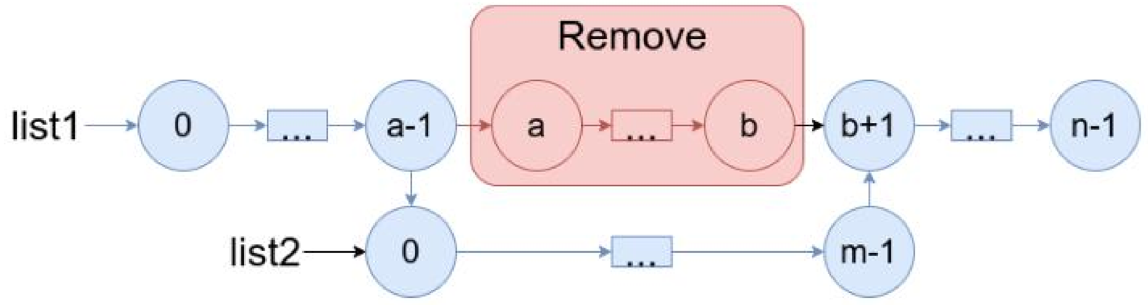
Input: head = [2,1]
Output: [2]

Explanation:
The above figure represents the given linked list.
For n = 2, node 1 with value 1 is the middle node, which is marked in red.
Node 0 with value 2 is the only node remaining after removing node 1.

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def deleteMiddle(self, head: ListNode) → ListNode:
        if not head or not head.next:
            return None

        slow = head
        fast = head
        prev = None

        while fast and fast.next:
            prev = slow
            slow = slow.next
            fast = fast.next.next

        prev.next = slow.next
        return head

# Helper functions to create and display the linked list
def create_linked_list(arr):
    head = ListNode(arr[0])
    current = head
    for val in arr[1:]:
        current.next = ListNode(val)
        current = current.next
    return head

def display_linked_list(head):
    current = head
    while current:
        print(current.val, end=" → ")
        current = current.next
    print("None")

# Example usage
head = create_linked_list([1, 3, 4, 7, 1, 2, 6])
sol = Solution()
new_head = sol.deleteMiddle(head)
display_linked_list(new_head)
```

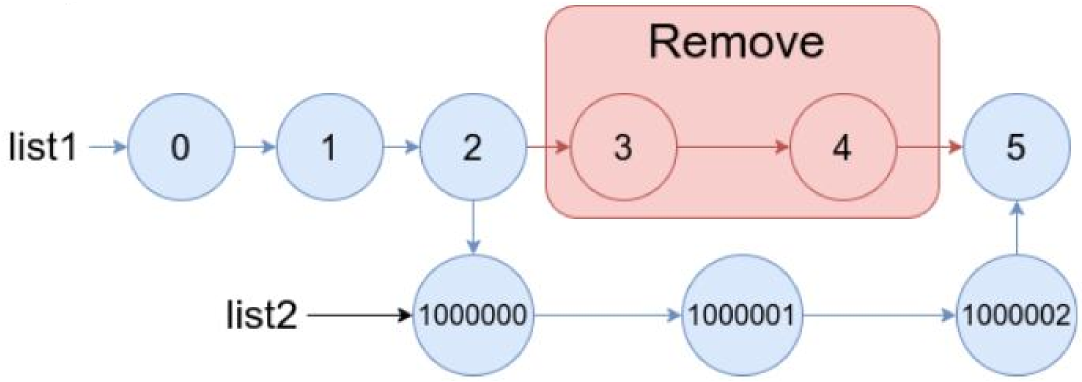
Q2.You are given two linked lists: list1 and list2 of sizes n and m respectively. Remove list1 's nodes from the ath node to the bth node, and put list2 in their place. [Leetcode 1669]

The blue edges and nodes in the following figure indicate the result:



Build the result list and return its head.

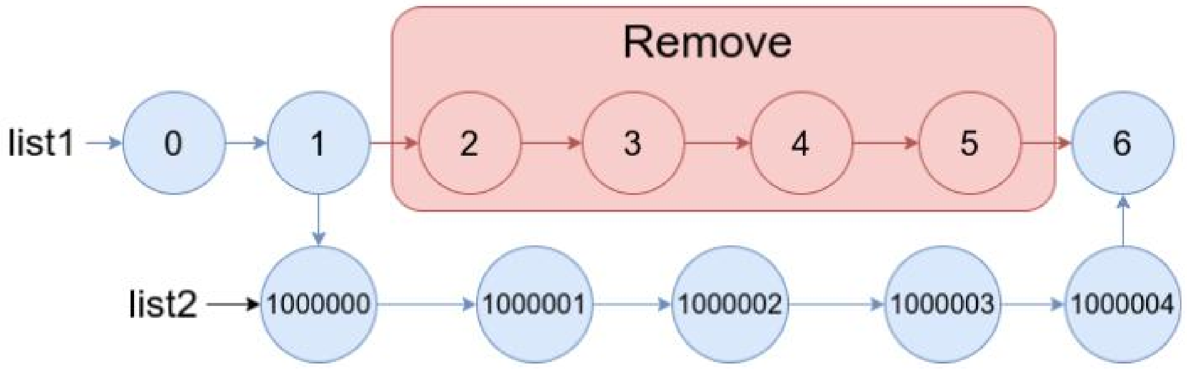
Example 1



Input: list1 = [0,1,2,3,4,5], a = 3, b = 4, list2 = [1000000,1000001,1000002]
Output: [0,1,2,1000000,1000001,1000002,5]

Explanation: We remove the nodes 3 and 4 and put the entire list2 in their place. The blue edges and nodes in the above figure indicate the result.

Example 2:



Input: list1 = [0,1,2,3,4,5,6], a = 2, b = 5, list2 =
[1000000,1000001,1000002,1000003,1000004]
Output: [0,1,1000000,1000001,1000002,1000003,1000004,6]

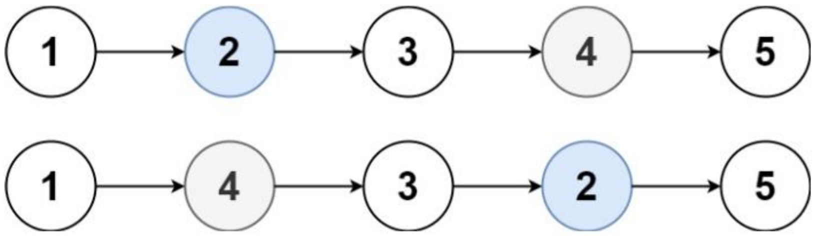
Explanation: The blue edges and nodes in the above figure indicate the result.

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def mergeInBetween(self, list1: ListNode, a: int, b: int,
list2: ListNode) -> ListNode:
        current = list1
        for _ in range(a - 1):
            current = current.next
        node_before_a = current

        for _ in range(b - a + 2):
            current = current.next
        node_after_b = current

        node_before_a.next = list2
 while list2.next:
            list2 = list2.next
        list2.next = node_after_b

        return list1

# Example usage
def create_linked_list(arr):
    head = ListNode(arr[0])
    current = head
    for val in arr[1:]:
        current.next = ListNode(val)
   current = current.next
    return head

def display_linked_list(head):
    current = head
    while current:
        print(current.val, end=" -> ")
        current = current.next
    print("None")

list1 = create_linked_list([0, 1, 2, 3, 4, 5])
list2 = create_linked_list([1000000, 1000001, 1000002])
a, b = 3, 4
sol = Solution()
new_head = sol.mergeInBetween(list1, a, b, list2)
display_linked_list(new_head)
```

3. You are given the head of a linked list, and an integer k .

Return the head of the linked list after swapping the values of the kth node from the beginning and the kth node from the end (the list is 1-indexed). [Leetcode 1721]

Example 1:



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

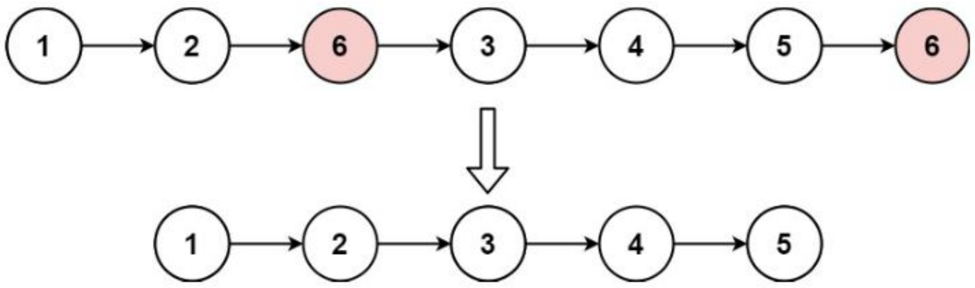
Example 2:
Input: head = [7,9,6,6,7,8,3,0,9,5], k = 5
Output: [7,9,6,6,8,7,3,0,9,5]

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
def getSize(self, head: ListNode) → int:
        size = 0
        current = head
        while current:
            size += 1
            current = current.next
        return size

    def swapNodes(self, head: ListNode, k: int) → ListNode:
        size = self.getSize(head)
        start_node = head
        end_node = head

        for _ in range(k - 1):
            start_node = start_node.next

        for _ in range(size - k):
            end_node = end_node.next

        start_node.val, end_node.val = end_node.val,
start_node.val
        return head
```

```
# Example usage
def create_linked_list(arr):
    head = ListNode(arr[0])
    current = head
    for val in arr[1:]:
        current.next = ListNode(val)
        current = current.next
    return head

def display_linked_list(head):
    current = head
    while current:
        print(current.val, end=" -> ")
        current = current.next
    print("None")

head = create_linked_list([1, 2, 3, 4, 5])
k = 2
sol = Solution()
new_head = sol.swapNodes(head, k)
display_linked_list(new_head)
```

4. Given the head of a linked list and an integer val , remove all the nodes of the linked list that has Node.val == val , and return the new head.

Example 1:



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

Example 2:
Input: head = [], val = 1
Output: []

Example 3:
Input: head = [7,7,7,7], val = 7
Output: []

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def removeElements(self, head: ListNode, val: int) ->
ListNode:
        dummy = ListNode(next=head)
        current = dummy

while current.next:
            if current.next.val == val:
                current.next = current.next.next
            else:
                current = current.next

        return dummy.next

# Example usage
def create_linked_list(arr):
if not arr:
        return None
    head = ListNode(arr[0])
    current = head
    for val in arr[1:]:
        current.next = ListNode(val)
        current = current.next
    return head

def display_linked_list(head):
    current = head
    while current:
        print(current.val, end=" -> ")
        current = current.next
    print("None")

# Example 1
head1 = create_linked_list([1, 2, 6, 3, 4, 5, 6])
sol = Solution()
new_head1 = sol.removeElements(head1, 6)
display_linked_list(new_head1)

# Example 2
head2 = create_linked_list([])
new_head2 = sol.removeElements(head2, 1)
display_linked_list(new_head2)

# Example 3
head3 = create_linked_list([7, 7, 7, 7])
new_head3 = sol.removeElements(head3, 7)
display_linked_list(new_head3)
```

5. Find the length of loop in Cycle of Linked List.

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def findLoopLength(self, head: ListNode) -> int:
        slow = head
        fast = head
  while fast and fast.next:
            slow = slow.next
            fast = fast.next.next
            if slow == fast:
                return self.countNodesInLoop(slow)

        return 0

    def countNodesInLoop(self, meetingPoint: ListNode) -> int:
        count = 1
        current = meetingPoint.next
        while current != meetingPoint:
            count += 1
            current = current.next
        return count

    def createLoop(self, head: ListNode, position: int):
        tail = head
        while tail.next:
            tail = tail.next

        loop_start = head
        for _ in range(position):
            loop_start = loop_start.next

        tail.next = loop_start

# Helper function to create and display the linked list
def create_linked_list(arr):
    head = ListNode(arr[0])
    current = head
    for val in arr[1:]:
        current.next = ListNode(val)
        current = current.next
    return head

def display_linked_list(head):
    current = head
    for _ in range(10):  # Limiting to 10 nodes to avoid infinite
loop in display
        if not current:
            break
        print(current.val, end=" -> ")
        current = current.next
    print("...")

# Example usage
head = create_linked_list([1, 2, 3, 4, 5])
sol = Solution()
sol.createLoop(head, 2)
print("Length of the loop:", sol.findLoopLength(head))
```