Bhagyashri Sutar

Roll No:-75

Assignment No:-03

Example 1

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

def rotateRight(head, k):
    """
    Rotates a linked list to the right by k places.

    Args:
        head: The head of the linked list.
        k: The number of places to rotate the list.

    Returns:
        The head of the rotated linked list.
    """

    if not head or k == 0:
        return head

    #1. Calculate the length of the linked list
    length = 1
    tail = head
    while tail.next:
        tail = tail.next
        length += 1

    # 2. Optimize k to avoid unnecessary rotations
    k = k % length

    if k == 0:
        return head  # No rotation needed

    # 3. Find the new tail and new head
    new_tail_pos = length - k - 1
    new_tail = head
    for _ in range(new_tail_pos):
        new_tail = new_tail.next

    new_head = new_tail.next
```

```
    # 4. Perform the rotation
    new_tail.next = None
    tail.next = head

    return new_head

# Helper function to create a linked list from a list
def create_linked_list(arr):
    if not arr:
        return None
    head = ListNode(arr[0])
    curr = head
    for i in range(1, len(arr)):
        curr.next = ListNode(arr[i])
        curr = curr.next
    return head

# Helper function to convert a linked list to a list
def linked_list_to_list(head):
    result = []
    curr = head
    while curr:
        result.append(curr.val)
        curr = curr.next
    return result

# Example usage
input_list = [1, 2, 3, 4, 5]
k = 2

head = create_linked_list(input_list)
rotated_head = rotateRight(head, k)
output_list = linked_list_to_list(rotated_head)

print(f"Input: {input_list}, k = {k}")
print(f"Output: {output_list}")

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

Example 2

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

def rotateRight(head, k):
    """
    Rotates a linked list to the right by k places.
```

```
    Args:
        head: The head of the linked list.
        k: The number of places to rotate the list.

    Returns:
        The head of the rotated linked list.
    """

    if not head or k == 0:
        return head

    # 1. Calculate the length of the linked list
    length = 1
    tail = head
    while tail.next:
        tail = tail.next
        length += 1

    # 2. Optimize k to avoid unnecessary rotations
    k = k % length

    if k == 0:
        return head  # No rotation needed

    # 3. Find the new tail and new head
    new_tail_pos = length - k - 1
    new_tail = head
    for _ in range(new_tail_pos):
        new_tail = new_tail.next

    new_head = new_tail.next

    # 4. Perform the rotation
    new_tail.next = None
    tail.next = head

    return new_head

# Helper function to create a linked list from a list
def create_linked_list(arr):
    if not arr:
        return None
    head = ListNode(arr[0])
    curr = head
    for i in range(1, len(arr)):
        curr.next = ListNode(arr[i])
        curr = curr.next
    return head
```

```python
# Helper function to convert a linked list to a list
def linked_list_to_list(head):
    result = []
    curr = head
    while curr:
        result.append(curr.val)
        curr = curr.next
    return result

# Example usage (Example 2)
input_list = [0, 1, 2]
k = 4

head = create_linked_list(input_list)
rotated_head = rotateRight(head, k)
output_list = linked_list_to_list(rotated_head)

print(f"Input: {input_list}, k = {k}")
print(f"Output: {output_list}")
```

```
Input: [0, 1, 2], k = 4
Output: [2, 0, 1]
```