Python Assignment no 3

By Parth Gawad, Roll no 62

## Example 1

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

def rotateRight(head, k):
    if not head or not head.next or k == 0:
        return head

    length = 1
    tail = head
    while tail.next:
        tail = tail.next
        length += 1

    k = k % length
    if k == 0:
        return head

    new_tail = head
    for _ in range(length - k - 1):
        new_tail = new_tail.next

    new_head = new_tail.next
    new_tail.next = None
    tail.next = head

    return new_head

def list_to_linked_list(lst):
    if not lst:
        return None
    head = ListNode(lst[0])
    current = head
    for val in lst[1:]:
        current.next = ListNode(val)
        current = current.next
    return head

def linked_list_to_list(head):
    result = []
    while head:
        result.append(head.val)
```

```
        head = head.next
    return result

head = list_to_linked_list([1, 2, 3, 4, 5])
k = 2
rotated_head = rotateRight(head, k)
print(linked_list_to_list(rotated_head))
```

```
[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):
    if not head or not head.next or k == 0:
        return head

    length = 1
    tail = head
    while tail.next:
        tail = tail.next
        length += 1

    k = k % length
    if k == 0:
        return head

    new_tail = head
    for _ in range(length - k - 1):
        new_tail = new_tail.next

    new_head = new_tail.next
    new_tail.next = None
    tail.next = head

    return new_head

def list_to_linked_list(lst):
    if not lst:
        return None
    head = ListNode(lst[0])
    current = head
    for val in lst[1:]:
        current.next = ListNode(val)
        current = current.next
    return head
```

```
def linked_list_to_list(head):
    result = []
    while head:
        result.append(head.val)
        head = head.next
    return result

head = list_to_linked_list([0, 1, 2])
k = 4
rotated_head = rotateRight(head, k)
print(linked_list_to_list(rotated_head))  # Output: [2, 0, 1]
```

```
[2, 0, 1]
```