# ARTIFICAL INTELLIGENCE

BY UZAIR KHAN
(BIT-24S-020)

# 61. ROTATE LIST

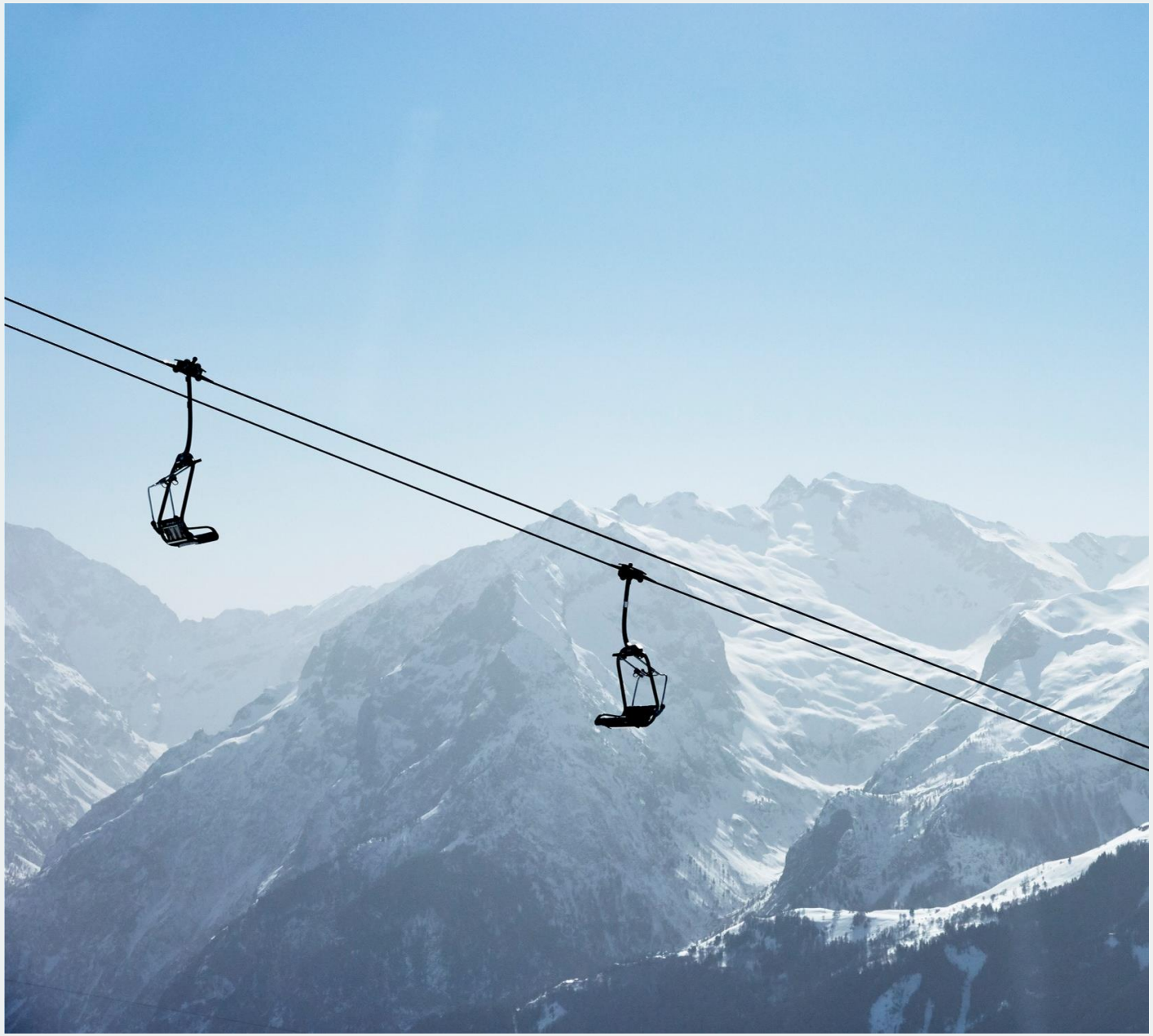
- Problem Statement:

Given the head of a linked list, rotate the list to the right by **k** places.

- Explanation:

If we are to rotate the list to the right by **k** steps, that means:

- The last **k** nodes of the list become the first **k** nodes.
- The list is treated circularly, then "cut" at the right point.



# PYTHON CODE (WITH DEF AND COMMENTS):

```
# Definition for singly-linked list.
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def rotateRight(self, head: ListNode, k: int) -> ListNode:
        if not head or not head.next or k == 0:
            return head

        # Step 1: Count length
        length = 1
        current = head
        while current.next:
            current = current.next
            length += 1

        # Step 2: Make the list circular
        current.next = head

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

        # Step 4: Break the circle
        new_head = new_tail.next
        new_tail.next = None

        return new_head
```

- EXAMPLE:

INPUT:

```
PYTHON
COPYEDIT
HEAD = [1, 2, 3, 4, 5]
K = 2
```

OUTPUT:

```
PYTHON
COPYEDIT
[4, 5, 1, 2, 3]
```

VISUAL FLOW:

ORIGINAL → 1 → 2 → 3 → 4 → 5
ROTATE 2 → RESULT → 4 → 5 → 1 → 2 → 3

# CONCLUSION

- The problem is about **circular rotation** of linked lists.
- Requires basic knowledge of **linked list traversal**.
- Key logic is making it circular and then **breaking at the correct point**.
- Edge cases: empty list, single node, or `k = 0`.