

Rotate list by k C++

```
#include <iostream>
```

```
struct Node {
    int val;
    Node* next;
    Node(int x) {
        val = x;
        next = nullptr;
    }
};
```

```
Node* rotateRight(Node* head, int k) {
    if (head == nullptr || k == 0) return head;
```

```
    int length = 1;
    Node* tail = head;
    while (tail->next != nullptr) {
        tail = tail->next;
        length++;
    }
```

```
    k = k % length;
    if (k == 0) return head;
```

```
    Node* newTail = head;
    for (int i = 0; i < length - k - 1; i++) {
        newTail = newTail->next;
    }
```

```
    Node* newHead = newTail->next;
    newTail->next = nullptr;
    tail->next = head;
```

```
    return newHead;
}
```

```
void printList(Node* head) {
    while (head != nullptr) {
        std::cout << head->val << " -> ";
        head = head->next;
    }
    std::cout << "null" << std::endl;
}
```

```
int main() {
    Node* head = new Node(1);
    head->next = new Node(2);
    head->next->next = new Node(3);
    head->next->next->next = new Node(4);
    head->next->next->next->next = new Node(5);
```

```
    Node* result = rotateRight(head, 2);
    printList(result);
```

```
    // Free the allocated memory
    Node* curr = result;
    while (curr != nullptr) {
        Node* temp = curr;
        curr = curr->next;
        delete temp;
    }
}
```

Problem Summary:

Rotate a singly linked list **to the right** by k places.

Input:

Linked List:

```
rust
CopyEdit
1 -> 2 -> 3 -> 4 -> 5
```

Rotate by k = 2

Dry Run Steps:

Step	Explanation	State
1	Initial list	1 -> 2 -> 3 -> 4 -> 5 -> null
2	Traverse list to find length and tail	length = 5, tail = 5
3	Normalize k: k = k % length = 2 % 5 = 2	Effective rotation is 2 places
4	Move to new tail: length - k - 1 = 5 - 2 - 1 = 2	Move 2 steps from head: node with value 3 is new tail
5	newTail = 3, newHead = 4, break link	newTail->next = nullptr, tail->next = head
6	New list after rotation	4 -> 5 -> 1 -> 2 -> 3 -> null

Final State:

- **Old Tail:** Node with value 5
- **Old Head:** Node with value 1
- **New Head:** Node with value 4
- **New Tail:** Node with value 3

Output:

4 -> 5 -> 1 -> 2 -> 3 -> null

```
return 0;  
}
```

Output:-
4 -> 5 -> 1 -> 2 -> 3 -> null