Problem 2: Rotate Elements of a Queue

Given a queue of integers `q` (e.g., `[1, 2, 3, 4, 5]`) and an integer `k` (e.g., `2`), write a function to rotate the elements of the queue by `k` positions to the right. This rotation should move each element in the queue `k` positions towards the back, with the elements at the end wrapping around to the front.

For example, if `q = [1, 2, 3, 4, 5]` and `k = 2`, after the rotation, the queue should be `[3, 4, 5, 1, 2]`.

Constraints:

- `1 ≤ k ≤ length of queue`

Hint: *Think about how dequeuing and enqueuing elements can change their order in the queue.*

Problem 2: Solution

```
void rotateQueue(queueLinkedList& queue, int k) {
    if (queue.isEmpty() || k <= 0) {
        std::cout << "Invalid operation." << std::endl;
        return;
    }

    int rotationCount = k % queue.size(); // to handle cases where k >
size of the queue

    for (int i = 0; i < rotationCount; i++) {
        int temp = queue.dequeue();
        queue.enqueue(temp);
    }
}
```

Class implementation which I used to write the code for the solution.

```
#include <iostream>

class queueLinkedList {
    class Node {
    public:
        int value;
        Node* next;
```

```
        Node(int value, Node* next = nullptr) : value(value), next(next)
{}
    };

    Node* front;
    Node* rear;
    int count;

public:
    queueLinkedList() : front(nullptr), rear(nullptr), count(0) {}

    ~queueLinkedList() {
        while (front != nullptr) {
            Node* temp = front;
            front = front->next;
            delete temp;
        }
    }

    void enqueue(int value) {
        Node* newNode = new Node(value);
        if (rear == nullptr) {
            front = rear = newNode;
        } else {
            rear->next = newNode;
            rear = newNode;
        }
        count++;
    }

    int dequeue() {
        if (front == nullptr) {
            std::cout << "Queue is empty" << std::endl;
            return -1; // Or throw an exception
        }

        Node* temp = front;
        int value = temp->value;
        front = front->next;
        if (front == nullptr) {
            rear = nullptr;
        }
        delete temp;
        count--;
        return value;
    }

    void display() {
        Node* current = front;
```

```
        while (current != nullptr) {
            std::cout << current->value << " -> ";
            current = current->next;
        }
        std::cout << "End" << std::endl;
    }

    bool isEmpty() const {
        return count == 0;
    }

    int size() const {
        return count;
    }
};

int main() {
    queueLinkedList queue;
    queue.enqueue(1);
    queue.enqueue(2);
    queue.enqueue(3);
    queue.enqueue(4);
    queue.enqueue(5);

    std::cout << "Original queue: ";
    queue.display();

    rotateQueue(queue, 2);

    std::cout << "Queue after rotating by 2 positions: ";
    queue.display();

    return 0;
}
```