Implement Queue Using Linked List CodeStudio

This program implements a queue data structure using a linked list. It defines a **Queue** class with methods for enqueuing, dequeuing, checking if the queue is empty, getting the front element, getting the size of the queue, and displaying its elements. The program demonstrates the usage of the **Queue** class by performing various queue operations.

1. **Node(int val)**: Constructor for the **Node** class to create a new node.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
2. **Queue()**: Constructor initializes an empty queue.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
3. **void enQueue(int val)**: Enqueues an element to the rear of the queue using a linked list.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
4. **int deQueue()**: Dequeues an element from the front of the queue using a linked list.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
5. **bool isEmpty()**: Checks if the queue is empty.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
6. **int getFront()**: Retrieves the front element of the queue without dequeuing it.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
7. **int getSize()**: Returns the size (number of elements) of the queue.
    - **Time Complexity: O(N), where N is the number of elements in the queue.**
    - **Space Complexity: O(1)**
8. **void display()**: Displays the elements in the queue.
    - **Time Complexity: O(N), where N is the number of elements in the queue.**

- **Space Complexity: O(1)**

9. **~Queue()**: Destructor to free the dynamically allocated memory for the linked list nodes.
    - **Time Complexity: O(N), where N is the number of elements in the queue.**
    - **Space Complexity: O(1)**