Implement Queue using Array CodeStudio

This program implements a basic queue data structure using an array. The **Queue** class has methods for enqueuing, dequeuing, getting the front and rear elements, checking if the queue is empty or full, 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. **Queue(int size)**: Constructor initializes the queue with the given size.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(N), where N is the size of the queue.**
2. **void enQueue(int val)**: Enqueues an element to the rear of the queue.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
3. **int deQueue()**: Dequeues an element from the front of the queue.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
4. **int getFront()**: Retrieves the front element of the queue without dequeuing it.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
5. **int getRear()**: Retrieves the rear element of the queue without dequeuing it.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
6. **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)**
7. **bool isEmpty()**: Checks if the queue is empty.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
8. **bool isFull()**: Checks if the queue is full.
    - **Time Complexity: O(1)**

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

9. **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)**
10. **~Queue()**: Destructor to free the dynamically allocated memory for the array.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**