Implement Queue Using Two Stacks LeetCode

This program implements a queue data structure using two stacks. It defines a **Queue** class with methods for pushing elements into the queue, popping the front element from the queue, getting the front element, getting the size of the queue, and checking if the queue is empty. The program demonstrates the usage of the **Queue** class by performing various queue operations.

1. **Queue()** - Constructor to initialize the queue.
   - **Time Complexity: O(1)**
   - **Space Complexity: O(1)**
   - **Explanation:** This constructor initializes the queue with two empty stacks, **st1** and **st2**. It's a simple operation with constant time and space complexity.
2. **void push(int x)** - Pushes an element into the queue using two stacks.
   - **Time Complexity: O(N), where N is the number of elements in the stack st1.**
   - **Space Complexity: O(N), where N is the number of elements in the queue.**
   - **Explanation:** This method transfers elements from **st1** to **st2**, pushes the new element onto **st1**, and transfers elements back from **st2** to **st1** to maintain the order. Since it involves a fixed number of operations regardless of the number of elements, the time complexity is constant. However, the space complexity is linear with respect to the number of elements in the queue due to the temporary storage in **st2**.
3. **int pop()** - Pops the front element from the queue.
   - **Time Complexity: O(1)**
   - **Space Complexity: O(1)**
   - **Explanation:** This method simply pops the top element from **st1**, which represents the front element of the queue. It has constant time and space complexity.
4. **int getFront()** - Retrieves the front element of the queue without dequeuing it.
   - **Time Complexity: O(1)**
   - **Space Complexity: O(1)**
   - **Explanation:** This method returns the top element of **st1**, which represents the front element of the queue. It has constant time and space complexity.
5. **int getSize()** - Returns the size (number of elements) of the queue.

- **Time Complexity: O(1)**
- **Space Complexity: O(1)**
- **Explanation:** This method returns the size of **st1**, which directly represents the size of the queue. It has constant time and space complexity.

6. **bool empty()** - Checks if the queue is empty.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
    - **Explanation:** This method checks if **st1** is empty, which indicates whether the queue is empty. It has constant time and space complexity.