# Implement Queue Using One Stack LeetCode

This program implements a queue data structure using a single stack. 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 an empty queue. It's a simple operation with constant time and space complexity.
2. **void push(int x)** - Pushes an element into the queue using a single stack.
    - **Time Complexity: O(N), where N is the number of elements in the stack.**
    - **Space Complexity: O(N)**
    - **Explanation:** This method pushes an element into the queue using a recursive approach. It involves popping elements from the stack until it's empty, pushing the new element to the bottom, and then pushing the previously popped elements back. The time and space complexity are both linear with respect to the number of elements in the stack.
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 the stack, which is the front element of the queue. It has a 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 the stack, 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 the stack, 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 the stack is empty, which indicates whether the queue is empty. It has constant time and space complexity.