Stack Using Two Queues LeetCode

This C++ program demonstrates the implementation of a stack using two queues. The **Stack** class is defined with two **queue** objects, **q1** and **q2**, to simulate the behavior of a stack. The program utilizes the property of queues where the front of the queue is treated as the top element of the stack.

The **Stack** class is defined with two private **queue<int>** objects, **q1** and **q2**.

1. **push (void push(int value)):**
    - Function Explanation: Pushes the given element onto the stack. It transfers elements from **q1** to **q2**, pushes the new element onto **q2**, swaps the queues, making **q2** the new empty queue, and **q1** the stack with the new element on top.
    - **Time Complexity: O(n), where n is the number of elements in the stack. Transferring elements from one queue to another takes linear time.**
    - **Space Complexity: O(1)**
2. **pop (int pop()):**
    - Function Explanation: Removes and returns the top element from the stack (front of **q1**).
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
3. **getTop (int getTop()):**
    - Function Explanation: Returns the top element of the stack (front of **q1**) without removing it.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
4. **isEmpty (bool isEmpty()):**
    - Function Explanation: Checks if the stack is empty by checking if **q1** is empty.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
5. **getSize (int getSize()):**

- Function Explanation: Returns the number of elements in the stack (size of **q1**).
- **Time Complexity: O(1)**
- **Space Complexity: O(1)**

6. **Destructor (~Stack()):**
    - Function Explanation: Releases memory used by the queues by iteratively popping elements from **q1**.
    - **Time Complexity: O(n), where n is the number of elements in the stack.**
    - **Space Complexity: O(1)**