Names:	Mushinzimana claude
Renumber:	224015060

Questions. on Stack AND Queue THEORY

Q.3.Challenge – Reverse "QUEUESTACK" using a stack

Algorithm (step-by-step)

1. Create an empty stack.

2. For each character in the input string (left → right) push the character onto the stack.

3. Create an empty result string.

4. While the stack is not empty, pop a character and append it to the result string.

5. The result string is the reversed input.2) Challenge – Voting lines: Queue (FIFO) vs Stack (LIFO) algorithm + code + explanation

Goal: demonstrate which data structure preserves arrival order (fairness) and quantify how "out-of-order" the serving becomes.

Algorithm (step-by-step)

1. Take the list of arrivals (IDs in arrival order).

2. Queue method (FIFO): enqueue all arrivals, then repeatedly dequeue to get served queue.

3. Stack method (LIFO): push all arrivals, then repeatedly pop to get served stack.

4. Compare both served orders to the original arrivals.

5. Compute a simple unfairness metric: count the number of arrival-pairs (I, j) where I < j but the served order places I after j (i.e., inversion count relative to arrival order). More inversions = more unfairness.

Explanation (mapping to algorithm):

q = deque(arrivals) / dequeuing builds served queue in the same order as arrivals (FIFO). That models a fair voting line.

stack. Append(a) / popping builds served stack which is the reverse of arrival order (LIFO). That models an unfair policy where the last arrival is served first.

inversions(served) computes how many ordered pairs (imp) from the original arrival list end up served out-of-order. Zero inversions mean perfect preservation of arrival order.

The function returns both sequences and their inversion counts so you can compare fairness numerically.

reflections (theory only)

A. Why is a stack best for reversing sequences ?
   - ✓ A stack implements LIFO (Last In, First Out). If you push the sequence elements in their original order, the last element becomes the first available to pop – popping repeatedly returns elements in reverse order. This requires a single pass to push (O(n)), a single pass to pop (O(n)), and O(n) extra space for the stack. It's simple, robust, and maps directly to the reversal operation. (Alternative approaches exist – e.g., in-place swapping two-pointer technique on mutable arrays – those are fine when you can modify the data in place; the stack-based method is universal, works on immutable sequences, and is easy to reason about.)

B. Why does FIFO ensure fairness in elections?

- ✓ FIFO (First In, First Out) gives service in the same order people arrived. That prevents later arrivals from being served before earlier arrivals, making waiting times predictable and avoiding queue-jumping. In elections this is crucial to equal treatment: each voter who arrives earlier receives service earlier, minimizing arbitrary advantages. Real systems sometimes add controlled exceptions (priority lanes for accessibility, reserved slots), but the baseline fair policy for the general queue is FIFO.

2. Challenge Question (algorithm + code + explanation)

Q: Queue vs stack for managing voting lines. Which is correct?

Algorithm (step-by-step)

1. Collect arrivals in a list (order voters arrive).

2. Model voting line with a queue (FIFO): serve voters in the order they arrive.

3. Model it incorrectly with a stack (LIFO): serve the last arrival first.

4. Compare the two orders:

FIFO preserves fairness.

LIFO reverses order → unfair.

5. Conclude: use queue for voting lines.

Explanation:

FIFO serves voters in arrival order → fair.

LIFO reverses the order → unfair for a voting line.

3. Reflection Question (theory only)

Q: Why does FIFO ensure fairness in elections?

- ✓ FIFO means First In, First Out. The first person to arrive is the first to be served. This prevents latecomers from jumping ahead and ensures voters are treated equally. It makes waiting times predictable and upholds fairness in the democratic process.