Reflection Notes on Queue & stack

1. **Why is FIFO critical in hospitals for non-emergency patients?**

Key Idea:
FIFO (**First In, First Out**) ensures patients are attended to in the same order they arrive. For **non-emergency cases**, this method balances fairness, trust, and operational efficiency.

Details (with assignment references):

- **Fairness & Equity**
    - Everyone is treated equally based on arrival time.
    - Prevents queue jumping or favoritism.
    - *(Reference: In the assignment's **BK ATM Queue** and **Nyabugogo Bus Queue**, the first in line is always the first served — hospitals must work the same way to ensure justice.)*
- **Trust & Satisfaction**
    - Transparent order reduces anxiety.
    - Patients gain confidence in Rwanda's healthcare services.
    - *(Reference: Just like **queues in ATM systems**, patients trust the process because it is predictable and unbiased.)*
- **Operational Efficiency**
    - Easy for staff to manage waiting areas without chaos.
    - Prevents disputes and ensures peace in busy hospital environments.
    - *(Reference: Similar to **bus queues in Nyabugogo terminal**, FIFO organizes flow efficiently.)*
- **Ethical & Legal Standards**
    - Aligns with professional medical ethics and Rwanda's constitutional right to equal healthcare.

Conclusion:
FIFO in hospitals protects **fairness, trust, and order** — making healthcare delivery smooth, ethical, and socially acceptable.

2. **Why is Stack suitable for temporary storage in problem-solving?**

Key Idea:
A stack's **LIFO (Last In, First Out)** behavior makes it ideal for handling temporary data where the most recent task must be solved or undone first.

Details (with assignment references):

- **LIFO Principle**
    - Recently added data is retrieved first, matching natural problem-solving.
    - *(Reference: In the **Momo App navigation stack**, the last step "Confirm" is undone first when pressing back.)*
- **Temporary Storage & Undo Operations**
    - Allows easy reversal of steps or operations.
    - *(Reference: In the **UR Exam System stack**, answers are stored temporarily and popped out in reverse order of entry.)*
- **Backtracking & Recursion**
    - Perfect for solving nested problems like recursion and parsing.
    - *(Reference: In the **Bracket Balancing algorithm**, opening brackets are pushed and popped when matching closing brackets.)*
- **Memory Efficiency**
    - Direct access to the top element (O (1) complexity).
    - Frees memory automatically when data is popped.

Conclusion:
Stacks are suitable for **temporary storage** because they are **fast, reversible, and memory-efficient**, especially for tasks like undoing operations, recursive problem solving, and navigation.