Name: NIYIGABA JEANETTE
Reg No: 224004103
Module: Data Structures and Algorithms(DSA)
Department: BIT

1. **Operation:**

   Q1. How does this show the LIFO nature of stacks?
   Because the last step you entered in the MTN MoMo app is the first one to be removed when you press back. That is exactly Last in First Out(LIFO)

2. **Push/Pop (LIFO) in MTN MoMo app.**

- **Stack rule**: *Last In, First Out (LIFO)* → the most recent item/action is the first to be removed.
- **MTN MoMo example**: When filling payment details (e.g., phone number → amount → PIN), each step is like **pushing** onto a stack.
- If you press **back**, the app removes the *last step you added* (the PIN step before showing amount).

  This demonstrates LIFO: the most recent (last) action is undone first.
-

---

2. Operation: Pop (Undo) in UR Canvas navigation

- **Pop operation**: Removes the top element of the stack.
- **UR Canvas example**: When you go back in modules, the most recent page you opened is removed first.
  Just like a stack pop(), the system discards the **latest visited module/page** before showing the earlier ones.

---

3. Application: Push for Undo in BK Mobile Banking (History + Undo)

- Every transaction (deposit, transfer, payment) is **pushed** onto the history stack.
- If you make a mistake, an **undo function** would simply **pop** the last transaction (the

Q4: How can stacks ensure forms are correctly balanced?

Stacks check balance by pushing every opening symbol (like {, (, [or form sections) and popping when the matching closing symbol appears.

- If at the end the stack is empty → everything is correctly matched.
- If something is left unmatched → the form is unbalanced.

So in **Irembo forms**, stacks help make sure every field opened (e.g., a sub-form or section) is properly closed before submitting.

Q5: Push and Pop sequence

Sequence:

1. Push ("CBE notes") → [CBE notes]

2. Push (“Math revision”) → [CBE notes, Math revision]
3. Push(“Debate”) → [CBE notes, Math revision, Debate]
4. Pop () → removes “Debate” → [CBE notes, Math revision]
5. Push (“Group assignment”) → [CBE notes, Math revision, Group assignment]

T of stack = “Group assignment”

Q6: Undo with multiple Pops

If the student undoes **three actions**, the last three pushed items are removed.

Suppose their stack (answers) was:

[A1, A2, A3, A4, A5]

- Pop 1 → removes A5.
- Pop 2 → removes A4.
- Pop 3 → removes A3.

Remaining stack = [A1, A2]

So, after undoing, only the earlier answers stay.

Do you want me to also **draw a diagram (stack box**

Q7: How does a stack enable this retracing process (Rwanda air booking backtracking)?

Each step in the booking form is **pushed onto the stack** as the passenger fills it.

When the passenger presses **“back”**, the **top step is popped**, restoring the previous state.

This allows retracing in **reverse order of actions (LIFO)** — exactly how stacks work.

Q8: Show how a stack algorithm reverses the proverb "Umwana ni umutware".

Steps:

1. Push each word onto the stack:
    - Push("Umwana"), Push("ni"), Push("umutware")

      Stack top → "umutware"
2. Pop words one by one:
    - Pop → "umutware"
    - Pop → "ni"
    - Pop → "Umwana"

Result after popping: **"umutware ni Umwana"** (the proverb reversed).

Q9: Why does a stack suit DFS in Kigali Public Library better than a queue?

In **DFS (Depth First Search)**, the search goes **deep along one path before backtracking**.

- A **stack** is perfect because it stores paths in LIFO order: last shelf explored is the first to be checked/undone.
- A **queue** (FIFO) would spread **level by level (BFS)**, which is slower for deep exploration.

  Thus, stack suits DFS because it naturally supports "go deep, then backtrack."

Q10: Suggest a feature using stacks for transaction navigation (BK Mobile app).

Feature: **"Undo/Redo Transaction Navigation"**

- Each time a user opens a transaction, it is **pushed onto the stack**.
- Pressing **Back** pops the current transaction and shows the previous one.
- A **forward (redo) stack** could store popped transactions, letting users go forward again.

  This improves navigation history, similar to a web browser's back/forward feature.

---

part:2

A. Basics

1. Operation: Enqueue (add at rear), Dequeue (remove from front)

- **Example:** At a restaurant in Kigali, customers are served in order.
- **Q1: How does this show FIFO behavior?**

  Answer: FIFO stands for *First In, First Out*. The first customer to arrive is the first one to be served, and new arrivals join the line at the back. This exactly mirrors a queue: items (customers) are added at the rear (enqueue) and removed from the front (dequeue).

---

2. Operation: Dequeue (next item leaves first)

- **Example:** In a YouTube playlist, the next video plays automatically.
- **Q2: Why is this like a dequeue operation?**

  Answer: Videos are played in the order they were added. The first video added to the playlist plays first, and as each video finishes, it is removed from the "queue," making the next video the new front. This mimics the dequeue operation where the front item is removed first.

B. Application

3. Operation: Enqueue (job submission)

- **Example:** At RRA offices, people waiting to pay taxes form a line.
- **Q3: How is this a real-life queue?**

  Answer: Each taxpayer joins the line at the end (enqueue), and the person at the front is served first (dequeue). This physical line is a real-life illustration of a queue data structure.

---

4. Operation: Queue management

- **Example:** In MTN/Airtel service centers, SIM replacement requests are processed in order.
- **Q4: How does queue management work here?**

  Answer: Each request is logged and placed in the order received. The first request is handled first, ensuring fairness. This is exactly how a queue structure maintains the order of processing tasks.

. Operation: Sequence of Enqueue/Dequeue

- **Operations:**

  Enqueue("Alice"), Enqueue("Eric"), Enqueue("Chantal"), Dequeue(), Enqueue("Jean")
- **Q5: Who is at the front now?**

  Step-by-step:

  1. Queue: Alice → front
  2. Enqueue Eric → Alice, Eric
  3. Enqueue Chantal → Alice, Eric, Chantal
  4. Dequeue → Alice leaves → Eric, Chantal
  5. Enqueue Jean → Eric, Chantal, Jean

     Answer: Eric is at the front.

---

6. Operation: FIFO message handling

- **Example:** RSSB pension applications managed in arrival order
- **Q6: How does a queue ensure fairness?**

  Answer: Each application is added to the end of the queue. The first submitted application is processed first, and subsequent ones follow in order. This prevents skipping anyone, ensuring fair treatment for all applicants.

D. Advanced Thinking

7. Operation: Different queue types

- **Examples & Real-life mapping:**
  - **Linear queue:** People at a wedding buffet → served in order, no one re-enters the line.
  - **Circular queue:** Buses looping at Nyabugogo → after the last bus, the next comes again in a cycle.
  - **Deque (double-ended queue):** Boarding a bus from front or rear → people can join or leave from both ends.
- **Q7:** Each queue type maps to real-life situations by showing **how items/customers are added and removed** in specific patterns.

---

8. Operation: Enqueue orders, Dequeue when ready

- **Example:** Kigali restaurant, customers order food and are called when ready
- **Q8: How can queues model this process?**

  Answer: Orders are added to a queue as they are received (enqueue). When the kitchen

prepares an order, it is removed from the front (dequeue) and served. This ensures orders are delivered in the order received.

---

9. Operation: Priority queue

- **Example:** CHUK hospital, emergencies jump the line
- **Q9: Why is this a priority queue, not a normal queue?**
  Answer: In a normal queue, everyone is served strictly by arrival order (FIFO). In a priority queue, some items (emergency patients) can bypass the line because their priority is higher. The order depends on urgency, not just arrival time.

---

10. Operation: Enqueue/Dequeue matching system

- I see your list got cut off here. Usually this refers to **matching requests with responses** using queues, such as ticketing systems or online customer service