# DATA STRUCTURE

# BIT GROUP; 1 ASSIGNMENT

# REG NUMBER: 224012652

## Q1: How does this show the LIFO nature of stacks?

In the MTN MoMo app, each payment step (e.g., entering amount, selecting recipient, confirming) is added to a stack. Pressing "Back" removes the most recent step—exactly like a Pop operation. Since the Last In step is the First Out, this demonstrates LIFO behavior.

## Q2: Why is this action similar to popping from a stack?

In UR Canvas, navigating modules adds each visited page to a stack. Pressing "Back" removes the current (topmost) page and returns to the previous one—mirroring a **Pop** operation that removes the top element of a stack.

## Q3: How could a stack enable the undo function when correcting mistakes?

Each transaction in BK Mobile Banking is **Pushed** onto a stack. To undo a mistake, the app **Pops** the most recent transaction, reverting to the prior state—just like an undo feature in text editors.

## Q4: How can stacks ensure forms are correctly balanced?

When entering data with brackets or tags (e.g., in Irembo forms), every opening symbol (like "(" or "{") is **Pushed**. When a closing symbol appears, the stack is Popped—if it matches, the form is balanced. Mismatches or leftover symbols indicate errors.

## Q5: Which task is next (top of stack)?

Sequence: Push("CBE notes") → Push("Math revision") → Push("Debate") → Pop() → Push("Group assignment")

After Pop(), “Debate” is removed. The stack now (top to bottom): “Group assignment”, “Math revision”, “CBE notes”.

Answer: “Group assignment” is on top.

---

## Page 2: Part I – STACK (Logical & Advanced Thinking)

### Q6: Which answers remain in the stack after undoing?

If a student undoes 3 recent actions, the stack Pops the last 3 entries. Only the actions **before** those three remain. For example, if the stack was [A, B, C, D, E] (E on top), after 3 Pops, [A, B] remain.

### Q7: How does a stack enable this retracing process?

Each booking step in RwandAir (e.g., selecting flight, entering passenger info, payment) is **Pushed**. To go back, the app **Pops** the current step, returning to the previous one—allowing step-by-step backtracking via LIFO.

### Q8: Show how a stack algorithm reverses the proverb.

Proverb: “Umwana ni umutware”

- Push each word: Push(“Umwana”) → Push(“ni”) → Push(“umutware”)

- Stack (top to bottom): “umutware”, “ni”, “Umwana”

- Pop each: “umutware” → “ni” → “Umwana”

Result: **“umutware ni Umwana” (reversed word order).

#### Q9: Why does a stack suit this case better than a queue?

In DFS (Depth-First Search), you explore one path as deeply as possible before backtracking—exactly what a stack supports (LIFO). A queue (FIFO) would explore breadth-first, which is inefficient for deep, narrow searches like library shelves.

### Q10: Suggest a feature using stacks for transaction navigation.

BK Mobile could implement a “Recent Path” navigator: each viewed transaction is **Pushed**. Users press “Back” to Pop and return to the prior transaction—enabling intuitive, reversible browsing of history.

---

## Page 3: Part II – QUEUE (Basics & Application)

### Q1: How does this show FIFO behavior?

At a Kigali restaurant, the first customer to arrive is served first. New customers join the end of the line. This mirrors a queue: Enqueuer at rear, Dequeuer from front—**First In, First Out**.

### Q2: Why is this like *a d*equeuer operation?

In a YouTube playlist, videos are played in the order they were added. When one ends, the next (front of queue) starts automatically—just like Dequeuer, which removes and processes the front item.

### Q3: How is this a real-life queue?

At RRA offices, taxpayers line up in arrival order. Each new person **joins the rear** (Enqueue), and the officer serves the **front person** (Dequeue)—a perfect real-world FIFO queue.

### Q4: How do queues improve customer service?

Queues ensure fairness and order: MTN/Airtel centers process SIM requests in arrival sequence. This reduces chaos, prevents cutting in line, and gives customers predictable wait times—enhancing trust and efficiency.

### Q5: Who is at the front now?

After Dequeue(), “Alice” leaves Operations: Enqueue(“Alice”) → Enqueue(“Eric”) → Enqueue(“Chantal”) → Dequeue() → Enqueue(“Jean”)

. Queue: [“Eric”, “Chantal”, “Jean”]

Front: “Eric”

---

## Advanced Page 4: Part II – QUEUE (Logical & Thinking)

### Q6: Explain how a queue ensures fairness.

RSSB handles pension apps in arrival order. A queue guarantees that earlier applicants are processed before later ones—no one is skipped. This **FIFO discipline** ensures equal treatment and transparency.

.

- Linear queue Wedding buffet line—people join end, served from front; line ends when food runs out.
- Circular queue: Nyabugogo bus loop—buses return to start after last stop, reusing “slots” efficiently.
- Deque: Bus boarding from both doors—passengers can enter/exit front or rear, like a doubleended queue.

### Q8: How can queues model this process?

Orders are Enqueuer when placed. When food is ready, the kitchen Dequeuers

the oldest order and calls the customer. This ensures orders are fulfilled in the sequence they were received.

### Q9: Why is this a priority queue, not a normal queue?

At CHUK, critical patients (e.g., heart attacks) are treated before stable ones—even if they arrived later. A **priority queue** assigns urgency levels, unlike a normal FIFO queue.

## Q10: How would queues fairly match drivers and students?

Riders (drivers) and passengers (students) each wait in separate FIFO queues. The system matches the **front** of each queue: first waiting driver + first waiting student. This ensures fairness and minimizes wait time.

---