Part I- STACK

Basics

Q1: How does this show the *LIFO* nature of stacks?
In the MTN MoMo app, pressing *back* removes the most recent step you entered (the last one you added). This follows the Last In, First Out (LIFO) rule of stacks: the last item added is the first to be removed.

Q2: Why is this action similar to popping from a stack?
In UR Canvas, pressing *back* undoes the last navigation step. This is the same as a pop operation, where the top (last added) element is removed from the stack.

Application

Q3: How could a stack enable the *undo* function when correcting mistakes?
Each action is pushed onto the stack as it is performed. To undo, the system pops the most recent actions from the stack, restoring the state before those actions just like going back step by step.

Q4: How can stacks ensure forms are correctly balanced?
When filling forms, each opening bracket (or data entry start) is pushed onto the stack. Each closing bracket (or end of a field) must match and pop the top element. If the stack is empty at the end, the form is correctly balanced.

Logical

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

Push(“CBE notes”) → [CBE notes]

Push(“Math revision”) → [CBE notes, Math revision]

Push(“Debate”) → [CBE notes, Math revision, Debate]

Pop() → removes “Debate” → [CBE notes, Math revision]

Push(“Group assignment”) → [CBE notes, Math revision, Group assignment]

Top of stack = “Group assignment”

Q6: Which answers remain in the stack after undoing?
If a student undoes 3 actions (3 Pops):

Suppose the stack is [Action1, Action2, Action3, Action4, Action5].

Pop 1 → removes Action5

Pop 2 → removes Action4

Pop 3 → removes Action3

The remaining stack = [Action1, Action2]

Advanced Thinking

Q7: How does a stack enable this retracing process?
Each step in the RwandAir booking form is pushed onto the stack. When the passenger goes *back*, the system pops the last step, allowing retracing in reverse order (LIFO).

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

Push words: [“Umwana”] → [“Umwana”, “ni”] → [“Umwana”, “ni”, “umutware”]

Pop sequence: → “umutware”, “ni”, “Umwana”

Reversed proverb = “umutware ni Umwana”

Q9: Why does a stack suit this case better than a queue?
In DFS (Depth-First Search), the goal is to go deep along one path before backtracking. A stack stores nodes in LIFO order, so the most recent unexplored branch is visited first.
A queue would follow BFS (level order), not deep search.

Q10: Suggest a feature using stacks for transaction navigation.
A “Back/Undo transaction navigation” feature:

Each screen (transaction step) is pushed onto the stack.

When the user presses “back”, the last transaction screen is popped, letting them retrace transaction history step-by-step.

<u>Part II- QUEUE</u>

<u>Basics</u>

Q1: How does this show FIFO behavior?
At a restaurant, the first customer to enter is the first to be served. This is First In, First Out (FIFO), exactly like a queue.

Q2: Why is this like a dequeue operation?
In a YouTube playlist, the first video (front of the queue) is played and removed. The next video automatically becomes the new front. This is the dequeue process.

Q3: How is this a real-life queue?
At RRA offices, each taxpayer joins at the rear of the line (enqueue), and the first to arrive is served first (dequeue). This is a real-world queue system.

Q4: How do queues improve customer service?
Queues ensure fairness and order:

First come, first served.

Prevents jumping the line.

Helps manage large crowds efficiently, reducing disputes and confusion.

<u>Logical</u>

Q5: Who is at the front now?
Sequence:

Enqueue("Alice") → [Alice]

Enqueue("Eric") → [Alice, Eric]

Enqueue("Chantal") → [Alice, Eric, Chantal]

Dequeue() → removes "Alice" → [Eric, Chantal]

Enqueue("Jean") → [Eric, Chantal, Jean]

Front = Eric

Q6: Explain how a queue ensures fairness.
In RSSB pension applications, the queue serves people strictly in arrival order (FIFO). No one skips ahead, so it is fair: the first applicant to arrive is the first processed.

<u>**Advanced Thinking**</u>

Q7: Explain how each maps to real Rwandan life.

Linear Queue: Wedding buffet line → people join at the end, and food is served in order.

Circular Queue: Buses looping at Nyabugogo → when a bus completes a trip, it re-enters the line at the back, making the queue continuous.

Deque (Double-ended Queue): Boarding a bus from front/rear → passengers can enter or leave from either end, just like a deque allows enqueue/dequeue at both ends.

Q8: How can queues model this process?
At a Kigali restaurant:

Customers place orders → each order is enqueued.

When food is ready → the chef dequeues the order from the front and calls the customer.
This models FIFO order processing.

Q9: Why is this a priority queue, not a normal queue?
At CHUK hospital:

Normal patients follow FIFO (first come, first treated).

Emergency cases jump ahead and are treated first, regardless of arrival time.
This is a priority queue, where urgency determines order instead of arrival time.

Q10: How would queues fairly match drivers and students?
In a moto/e-bike taxi app:

Drivers are enqueued as they become available.

Students (passenger requests) are also placed in a queue.

Matching happens by dequeuing the front driver with the front student request, ensuring fairness and first-come, first-served order.

END.