Part I - STACK

A. Basics

Q1: Pressing back in the MTN MoMo app removes the most recently entered payment detail, mirroring LIFO where the last item pushed (added) is the first popped (removed).

Ex: Think about you go through these steps in the Momo app:

1. Entered phone number

2. Entered amount

3. Entered PIN. If you press back, you will remove the PIN entry last. The last in (PIN) is the first out.

Q2: Pressing back in UR Canvas removes the most recent navigation step from the top of the stack, just like popping, which undoes the last action while preserving earlier ones.

Ex: 1. You open Module A_ push ("module A")

2. Then open module B_ push ("module B")

3. Then open module C_ push ("module C")

Now you press back:

. Pop () _ removes "Module C"

. Pop () _ removes "Module B"

B. Application

Q3: Each transaction or action can be pushed onto a stack as it's performed. To undo a mistake, pop the top item(s) to revert to the previous state, allowing step-by-step correction without losing the entire history.

Ex. Push ("send 10000 rwf")

. Push ("Pay bill")

.Pop ("Pay bill")

.Now only “send 10000 rwf” remains

Q4: Stacks can push opening fields/brackets and pop them when matching closing ones are encountered. If the stack is empty at the end with no mismatches, the form is balanced; otherwise, it indicates an error.

📌 Example: Suppose a form has matching sections:

1. push ("Start personal info")

2. push ("Start address")

3. pop () → address ends

4. pop () → personal info ends

C. Logical

Q5: After the operations:

- Push ("CBE notes") → Stack: ["CBE notes"] (top: "CBE notes")

- Push ("Math revision") → Stack: ["CBE notes", "Math revision"] (top: "Math revision")

- Push("Debate") → Stack: ["CBE notes", "Math revision", "Debate"] (top: "Debate")

- Pop () → Stack: ["CBE notes", "Math revision"] (top: "Math revision")

- Push ("Group assignment") → Stack: ["CBE notes", "Math revision", "Group assignment"] (top: "Group assignment")

The next task (top of stack) is "Group assignment".

Q6: Assuming actions (e.g., answers) are pushed onto the stack as they occur, undoing 3 recent actions means popping the top 3 items. The remaining answers in the stack would be the earlier ones that were pushed before the last 3.

Suppose actions were:

Push ("Answered Q1")

Push ("Answered Q2")

Push ("Answered Q3")

Push ("Answered Q4")

Push ("Answered Q5")

Now, the student undoes 3 actions → 3 Pops.

✅ Answer:

Remaining answers in the stack:

"Answered Q1", "Answered Q2"

D. Advanced Thinking

Q7: Each forward step in the booking form is pushed onto the stack. Going back pops the top step, retracing the path in reverse order (LIFO), allowing the passenger to revisit previous states without losing context.

Q8: Proverb: "Umwana ni umutware" (words: "Umwana", "ni", "umutware").

- Push "Umwana" → Stack: ["Umwana"]

- Push "ni" → Stack: ["Umwana", "ni"]

- Push "umutware" → Stack: ["Umwana", "ni", "umutware"]

- Pop "umutware", Pop "ni", Pop "Umwana" → Reversed: "umutware ni Umwana".

This works because popping reverses the push order (LIFO).

Q9: DFS explores deeply down one path before backtracking, which matches searching library shelves (e.g., one section deeply). A stack supports this by pushing paths and popping to backtrack, while a queue (FIFO) would breadth-search superficially across shelves, which is less suitable.

Q10: A "recent transactions" viewer where each viewed transaction is pushed onto a stack. Users can "go back" by popping to return to previous views, or "forward" by re-pushing, enabling easy navigation through history like a browser's back/forward buttons.

Part II - QUEUE

A. Basics

Q1: Customers enter at the rear (enqueue) and are served from the front (dequeue), ensuring the first to arrive is served first, demonstrating FIFO behavior.

Q2: The playlist acts as a queue where videos are enqueued in order. The next video dequeues and plays automatically, removing it from the front like FIFO, maintaining playback sequence.

B. Application

Q3: People join at the rear (enqueue) and are served from the front (dequeue) in arrival order, forming a FIFO line that processes tax payments sequentially.

Q4: Queues process requests in FIFO order, reducing wait time predictability, preventing skipping, and ensuring fair, organized service, which minimizes chaos and improves satisfaction.

C. Logical

Q5: After the operations:

- Enqueue("Alice") → Queue: ["Alice"] (front: "Alice")
- Enqueue("Eric") → Queue: ["Alice", "Eric"] (front: "Alice")
- Enqueue("Chantal") → Queue: ["Alice", "Eric", "Chantal"] (front: "Alice")
- Dequeue () → Queue: ["Eric", "Chantal"] (front: "Eric")

- Enqueue("Jean") → Queue: ["Eric", "Chantal", "Jean"] (front: "Eric")

The front now is "Eric".

Q6: Queues Process applications FIFO, so the first submitted is handled first, ensuring no one jumps ahead unfairly based on status, promoting equal treatment by arrival time.

D. Advanced Thinking

Q7: - Linear queue: People join at the rear and leave from the front in a straight line at a wedding buffet, processing sequentially without reuse of space.

- Circular queue: Buses arrive and depart in a loop at Nyabugogo station, reusing the "front" as the "rear" wraps around for efficiency.

- Deque: Passengers board from the front or rear of a bus, allowing additions/removals from both ends for flexible entry/exit.

Q8: Orders are enqueued as they arrive. When ready, the front order is dequeued and the customer called, modeling FIFO to handle preparation in sequence while notifying in arrival order.

Q9: Emergencies are given higher priority and inserted ahead (not strictly FIFO), allowing them to "jump" the line based on urgency, unlike a normal queue where everyone waits in arrival order.

Q10: Drivers enqueue in a queue upon availability. Passengers request, and the front driver is dequeued and matched, ensuring the longest-waiting driver gets the next ride for fair distribution.