date: 9/24/2025

COLLAGE: CBE

DEPARTMENT: BIT

MODULE: DATABASE MABAGEMENT

LECTURER: Dr prince

NAME: felix NIYONZIMA

REG NUMBER: 224010510

DATE: 09/24/2025

ASSIGNMENT OF DATABASE MANAGEMENT MANAGENT SYSTEM

Q1. MTN MoMo App Example (Push/Pop - LIFO)

Operation: Push/Pop (Last In, First Out)

Explanation:

- In a stack, the last item added is the first to be removed.
- In the MTN MoMo app, when you fill payment details step by step, each step is "pushed" onto a stack of steps.
- Pressing **back** removes the **most recent step** (last pushed), demonstrating LIFO behavior.
- Each step depends on previous steps, so you can only undo the most recent step first, just like popping from the top of a stack.

Q2) UR Canvas Example (Pop - Undo)

Operation: Pop

Explanation:

- In a stack, the **pop** operation removes the top item, which is the most recently added.
- In UR Canvas, navigating course modules adds each visited module to a temporary "ack".
- Pressing **back** removes the **last module visited**, similar to popping the top of a stack.

1. Order matters: You undo steps in reverse order; you cannot skip to earlier modules directly.
2. Temporary history: Each module visited is stored temporarily, like items in a stack.
3. Last action undone first: The most recent navigation is always undone first, demonstrating LIFO behavior.

Q3: Stack & Undo in BK Mobile Banking

- **Operation:** Push (Add to stack)
- **Explanation:**

  - Every transaction (or action) is **pushed onto the stack** of history.
  - If you make a mistake, the **undo function can pop the last transaction**, removing only the most recent one without affecting earlier ones.
  - This works because of **LIFO (Last In, First Out)**: the last action is always the first one that can be undone.

For example: 1 Push: Deposit 5,000

2 Push: Withdraw 2,000

3 Push: Transfer 1,000

Q4: Balanced Parentheses Check in Irembo Forms

- **Operation:** Push/Pop for matching brackets
- **Explanation:**
  - When you start filling forms, each **opening field** (like “start date”, “first name input”) is like an **opening bracket pushed onto the stack**.
  - When you complete and close the field properly, it is **popped from the stack**.
  - If at the end the stack is empty, all fields were correctly opened and closed (balanced).
  - If not, it means some fields were left incomplete or unmatched, showing an error.

Q5: Push and Pop Sequence

Steps:

1. Start → []
2. Push("CBE notes") → ["CBE notes"]
3. Push("Math revision") → ["CBE notes", "Math revision"]
4. Push("Debate") → ["CBE notes", "Math revision", "Debate"]
5. Pop() removes "Debate" → ["CBE notes", "Math revision"]
6. Push("Group assignment") → ["CBE notes", "Math revision", "Group assignment"]

SO, The task on top of the stack (next to do) is **"Group assignment**

Q6: Undo with Multiple Pops

Based on Q5 the remaining stack will be: stack: [ ] means empty

Example stack before undo:
["Answer1", "Answer2", "Answer3", "Answer4", "Answer5"]

- Undo 1 (Pop) → removes "Answer5"
- Undo 2 (Pop) → removes "Answer4"
- Undo 3 (Pop) → removes "Answer3"

Remaining stack: ["Answer1", "Answer2"]

Q7: Pop to backtrack (Rwandair booking)

- In the booking form, each step (e.g., passenger info → flight details → payment) is **pushed** onto a stack.
- If the passenger presses **back**, the app **pops the most recent step** first, showing the previous one.
- This enables **retracing step-by-step in reverse order**, just like popping items from a stack.

Q8: Reverse proverb using stack

Proverb: **“Umwana ni umutware”**

Algorithm:

1. Split into words → ["Umwana", "ni", "umutware"]
    - Push each word →Push("Umwana") → Stack = ["Umwana"]
    - Push("ni") → Stack = ["Umwana", "ni"]
    - Push("umutware") → Stack = ["Umwana", "ni", "umutware"]
    - Pop all words →
    - Pop → "umutware"
    - Pop → "ni"

- Pop → "Umwana"

Result: “umutware ni Umwana”

So, Reversing is possible because stacks follow **LIFO system**

Q9: DFS in Kigali Public Library (Stack vs Queue)

- **DFS (Depth-First Search)** goes deep into one branch (shelf → section → row → book) before backtracking.
- A **stack** is suitable because it remembers the **last location explored**, and when popping, it backtracks correctly.
- A **queue** would explore **level by level** (BFS), which is slower for deep searches like shelves in a library.
- **Reason:** Stacks naturally support the **go-deep-then-backtrack** process of DFS.

Q10: Push/Pop in BK Mobile app navigation

- Each time you open a transaction detail, it’s **pushed** onto the navigation stack.
- Going **back** pops the current detail and returns you to the previous screen.

Suggested feature:
A **“Quick Undo Transaction View”** button that lets users pop multiple steps at once (e.g., jump back 3 transactions), powered by stack operations.

PART 2

Q1: Shows FIFO because:

1. The first customer to arrive is served first
2. Everyone is served in the exact order they join the line

Q2: Like dequeue since:

1. The next video at the front of the playlist plays automatically
2. Videos are removed from the front in the order they were added

Q3: Real-life queue because:

1. People join the line in the order they arrive

2. Each person is served one at a time from the front

Q4: Queues improve customer service because:

1. Requests are handled fairly, without skipping anyone
2. Service becomes organized and efficient

Q5: The front is **Eric** because:

1. Alice, who was first, has been dequeued
2. Eric is now at the front after the sequence of enqueue and dequeue operations

Q6: Ensures fairness because:

1. Applications are handled in the order of arrival
2. No application can skip ahead, ensuring equal treatment

Q7: Real-life mapping:

1. Linear queue : buffet line, where people are served in order
2. Circular queue : buses looping at Nyabugogo; Deque = boarding from front or rear

Q8: Models restaurant orders because:

1. Orders are queued as they are placed (enqueue)
2. Orders are served in sequence when ready (dequeue)

Q9: Priority queue because:

1. Emergencies are served before normal cases
2. Service order depends on urgency, not just arrival time

Q10: Feature for BK Mobile app because:

1. Users can navigate transaction history by popping recent transactions
2. Stack ensures the most recent transactions are accessed first, improving usability