BIT - Data structure Exercise - number 2

Part I - STACK

Definition:

A Stack is a linear data structure where insertion and removal of elements follow the **LIFO** principle **Last In, First Out.**

- You can only add **(Push**) or remove (**Pop**) elements from the **top** of the stack.
- An example of stacking books or plates — the last one you add is the first one you remove.

A. Basics

Q1, In the MTN MoMo app, each step in a transaction is added (pushed) to a stack. When you press **Back**, the app removes (pops) the **last step** you performed.

This is LIFO: the **last step** you entered is the **first one** to be removed.

Q2, "**Undo**" removes your most recent action — just like a **Pop** operation removes the top item from a stack.
It follows LIFO: the **last thing you did** is the **first thing undone**.

B. Application

Q3, each transaction is pushed onto a stack.
If there's a mistake, the **Undo** button pops the last transaction, removing it from history.
This way, users can **undo actions in reverse order** — starting from the most recent.

Q4, when checking input like ( ) or { }, a stack stores each opening bracket.
When a closing bracket appears, the stack **pops** the last opening bracket.
If all brackets match correctly, the stack will be empty — meaning the form is **balanced and correctly filled**.

C. Logical

Q5, Push and pop sequence

```
Push("CBE notes")
Push("Math revision")
Push("Debate")
Pop()
Push("Group assignment")
```

On the top after all operations: ["CBE notes", "Math revision", "Group assignment"]

Top: Group assignment”

Q6, If 3 actions are undone (3 Pops), the remaining in the stack:

Initial Stack: ["CBE notes", "Math revision", "Debate", "Group assignment"]

After 3 pops: remove → “Group assignment”, “Debate”, and “Math revision”

Remaining: “CBE notes”

D. Advanced thinking

Stack in retracinggg steps in RwandaAir Booking:

Each form step is pushed into a stack.
When the user presses **back**, the app pops the last step, going backward one step at a time.
This allows **step-by-step retracing** — exactly how stacks work.

a stack reverse “Umwana ni umutware”

- Push each word: Push("Umwana"), Push("ni"), Push("umutware")
- Then Pop each word: "umutware", "ni", "Umwana"

Reversed: **“umutware ni Umwana”**

Q9. Reason why a stack better than a queue in DFS (Deep search):

DFS explores one path **deeply** before going back.
A **stack** stores the last visited location, so it can backtrack correctly.
While, A **queue**, on the other hand, explores level by level (not deep), which suits BFS (Breadth-First Search), not DFS.

Q10. Ways can stacks help in BK app transaction navigation:

If the user wants to **go back**, the app pops the last viewed transaction.
This makes it easy to **navigate back step-by-step**.

PART II - QUEUES

DEFINITION

A Queue is a linear data structure where insertion and removal follow **the FIFO** principle — **First In, First Out.**

- **Enqueue** adds an item at the **rear**.
- **Dequeue** removes an item from the **front**.

A. Basics

Q1. a restaurant queue show FIFO: Customers join the line (enqueue).
The first person in line is the first to be served (dequeue).
This is FIFO: **First In, First Out**.

Q2. A youtube playist like dequeue: Videos are **played in the order** they were added.
Each video is dequeued (removed from front) and played.

B. Application

Q3. A tax payment queue a real-life queue: People waiting to pay taxes are served in the order they arrive.
Each new person joins the end (enqueue), and the person at the front is served first (dequeue).

Q4. Queues improve service at MTN/Airtel centers: By serving customers in the order they arrive, queues ensure fairness.
They prevent people from **jumping the line**, making the process **organized and efficient**.

C. Logical

Q5. From the opretion:

```
Enqueue("Alice")
Enqueue("Eric")
Enqueue("Chantal")
Dequeue()
Enqueue("Jean")
```

The one who is the front is **Eric**

Q6. Yes, queues ensure fairness in processing applications: Everyone is added to the queue **in the order they apply**.
Applications are processed one-by-one from the front.
No one is skipped or delayed unfairly — **first come, first served**.

Q7. Types of queues in Rwandan context:

Types of queue	**Real-life examples**
Linear queue	People lining up at a buffet (serve in order).
Circular queue	Buses at stations that loop routes continuously.
Deque (Double Ended Queue)	Boarding a bus from both front and rear doors.

Q8. queues for model food ordering in restaurants: Orders are added to the queue (enqueue).
When food is ready, it is called out in the same order (dequeue).
This keeps the service **orderly and fair**.

Q9. Queue at CHUK: Some patients (emergencies) are more critical than others.
They **jump ahead** in the queue, based on urgency — not arrival time.
That's why it's a **priority queue**, not a normal FIFO queue.

Q10. Fairly match moto drivers with students:

- One for waiting **students**
- One for waiting **moto drivers**

Always match the **first student** with the **first available driver**.
This keeps the process **fair and balanced**.