NAMES: MUKESHIMANA Aline

REG NUMBER: 224012261

CLASS: BIT GROUP1

DATA SRTUCTURE AND ALGORITHM

PART1: STACK

A **stack** is a **linear data structure** that follows the **Last In, First Out (LIFO)** principle. This means that the **last element added** to the stack is the **first one to be removed**.

- **Main operations**:
    1. **Push** – Add an element to the top of the stack.
    2. **Pop** – Remove the top element from the stack.
    3. **Peek/Top** – View the top element without removing it.
    4. **IsEmpty** – Check if the stack has no elements.

Ex: a stack of books

Top -> [Book D]

[Book C]

[Book B]

Bottom -> [Book A]

- **Push ("Book E")** → adds Book E on top.
- **Pop ()** → removes Book D (the last one added).
- **Peek ()** → shows Book D without removing it.

A. BASICS

Q1: How does this show the LIFO nature of stacks?In the MTN MoMo app, each payment step is pushed onto a stack. Pressing "Back" removes (pops) the last step you took, showing that the **Last In is the First Out (LIFO)**

The last action is undone first.

Q2: Why is this action similar to popping from a stack?
Just like a stack removes the top item with a **pop**, going back in UR Canvas undoes the most recent navigation step. The app removes the latest (top) step, mimicking stack behavior.

B. APPLICATION

Q3: How could a stack enable the undo function when correcting mistakes?
Each transaction is **pushed** onto a stack. To correct mistakes, the system **pops** the most recent action, restoring the state before that transaction just like "undo" in a text editor.

Q4: How can stacks ensure forms are correctly balanced?
As users fill forms with matching elements (start/end fields), opening fields are **pushed**, and when a corresponding closing field is entered it is **popped**. If all items match and the stack is empty at the end, the form is **balanced**.

C. LOGICAL

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

Push ("CBE notes")

Push ("Math revision")

Push("Debate")

Pop () → removes "Debate"

Push ("Group assignment")

At the top of the stack is **Group assignment**

Q6: Which answers remain in the stack after undoing?

Assume1:

Push("Answer1"), Push("Answer2"), Push("Answer3"), Push("Answer4"), Push("Answer5")

Undo 3 actions: Pop (), Pop (), Pop ()

Remaining: **Answer1, Answer2**

Push("Answer1"), Push("Answer2"), Push("Answer3")

Undo 3recent actions: Pop (), Pop (), Pop ()

The stack is empty

D. ADVANCED THINKING

Q7: How does a stack enable this retracing process?

Each form step is **pushed** as the user progresses. To backtrack, the app **pops** the last step, allowing the user to return to the previous one. This step-by-step **reverse navigation** follows stack logic.

Q8: Show how a stack algorithm reverses the proverb.

Proverb: "Umwana ni umutware"

Step 1: Push each word:

Push("Umwana")

Push("ni")

Push("umutware")

Step 2: Pop to reverse:

Pop () → "umutware"

Pop () → "ni"

Pop () → "Umwana"

Output: "umutware ni Umwana"

Q9: Why does a stack suit this case better than a queue?
In **Depth-First Search (DFS)** like exploring library shelves, the user follows one path deeply before backtracking. A stack helps by pushing paths and **popping back** when a path ends, unlike a queue (used in breadth-first search), which explores all options level by level.

Q10: Suggest a feature using stacks for transaction navigation.
"Undo last 5 transactions" feature: Each transaction is pushed onto a stack. Users can press **Undo** to **pop** recent transactions and view the state before them. It could also offer **"Redo"** by using a second stack to temporarily store popped items.

PART2: QUEUE

A **queue** is a **linear data structure** that follows the **First In, First Out (FIFO)** principle. This means that the **first element added** to the queue is the **first one to be removed**.

Main operations:

1. **Enqueue**: Add an element to the rear (end) of the queue.
2. **Dequeue**: Remove the element from the front of the queue.
3. **Front/Peek**: View the front element without removing it.
4. **IsEmpty**: Check if the queue has no elements.

Ex; a queue at a bank:

Front -> [Customer A] [Customer B] [Customer C] <- Rear

- **Enqueue ("Customer D")** → adds to the rear.
- **Dequeue ()** → removes "Customer A" (first one in line).
- **Peek ()** → shows "Customer A" without removing it.

A. Basics

Q1: How does this show FIFO behavior?

In a restaurant in Kigali, customers are served **in the order they arrive**.

- The **first** person to enter the line is the **first** to be served.
- This is **First-In, First-Out (FIFO)** like a **queue**, where the **earliest entry is processed first**.

Q2: Why is this like a dequeue operation?

In a YouTube playlist, the **next video** is **automatically removed from the front** of the list to be played.

- This is like a **dequeue** operation where the item at the **front** of the queue is taken out.
- It follows FIFO: The video added first plays first.

B. Application

Q3: How is this a real-life queue?

At RRA offices, people **join a line** to pay taxes.

- Each person **Enqueues** by joining the line at the back.
- As officers become available, they **Dequeue** the next person (from the front).

Q4: How do queues improve customer service?

Queues:

- Ensure customers are **served in order**, avoiding confusion or jumping.
- Make the **process fair**.
- Help staff handle one request at a time, improving **efficiency** and **organization**.

C. Logical

Q5: Who is at the front now?

Given:

Enqueue(“Alice”)

Enqueue(“Eric”)

Enqueue(“Chantal”)

Dequeue () → removes **Alice**

Enqueue(“Jean”)
Front → **Eric**, Chantal, Jean

Output: Eric is at the front.

Q6: Explain how a queue ensures fairness.

RSSB handles pension applications in the **order they arrive**:

- The **first person to apply** is the **first to be processed**.
- No one can **skip the line**.
- This **FIFO queue** ensures **equal treatment** and fairness for all applicants.

D. Advanced Thinking

Q7: Explain how each maps to real Rwandan life.

- **Linear Queue (e.g., Wedding Buffet):**
  People line up in a **straight line** and are served **in order** FIFO. No one joins in the middle.
- **Circular Queue (e.g., Nyabugogo Buses):**
  Buses leave the terminal, loop through their routes, and return to the terminal to **repeat the cycle** like a **circular queue**, which loops around.
- **Deque (e.g., Bus Boarding from Front/Rear):**
  A **double-ended queue** allows people to enter or exit from **both ends**, similar to boarding or leaving a bus from **front or back doors**.

Q8: How can queues model this process?

In a Kigali restaurant:

- **Customers place orders**: each order is **enqueued**.
- As meals are cooked and ready: each is **dequeued**, and the customer's name is called.
- Ensures meals are served in the **same order as they were ordered**, maintaining **FIFO**.

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

At CHUK hospital:

- **Emergency patients** are treated **before** others, even if they arrived later.
- Unlike a normal FIFO queue, **patients are sorted by priority**, not just time of arrival.
- This is a **priority queue**, where **critical cases jump the line** based on urgency.

Q10: How would queues fairly match drivers and students?

In a moto/e-bike app:

- **Riders (drivers)** form a queue they **Enqueue** as they come online.

- **Students** also form a queue when requesting rides.
- The **first available rider** is **matched** with the **first student waiting** (FIFO for both).
- This avoids skipping or delays, ensuring both parties are served in the order they came.