UNIVERSITY of RWANDA

Collage: BUSINESS AND ECONOMICS

DEPARTMENT: BUSINESS IFORMATION TECHINOLOGY

GEG NUMBERS: 224002813

NAMES: UWAMAHIRWE VALENS

Module:DATA STRACTURE AND ARIGOTHM

BIT-Data Stracture Exercise-number2

1. Part I – STACK

A. Basics

Q1: How does this show the LIFO nature of stacks?

In the MTN MoMo app, when you fill in payment details step-by-step, each step is pushed onto a stack. Pressing "Back" removes the most recent step (the last one entered). This is LIFO—**Last In, First Out**—because the most recent (last) step is the first to be removed.

Q2: Why is this action similar to popping from a stack?

Pressing back in UR Canvas undoes the last navigation step, just like a **pop** operation in a stack, where the most recent item is removed. It reverses your last action, reflecting the **top of stack** being removed.

Q3: How could a stack enable the undo function when correcting mistakes?

Each action (like a transaction or a typing step) is **pushed** onto a stack. When a mistake occurs, the system can **pop** the last few actions to undo them, restoring the previous state. This provides a clear and efficient undo mechanism.

Q4: How can stacks ensure forms are correctly balanced

For every **opening bracket or section**, push it onto the stack. When a **closing bracket or section** is found, pop from the stack and check if it matches. If the stack is empty at the end and all matches were correct, the form is balanced—just like matching parentheses.

C. Logical

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

`Push("CBE notes"), Push("Math revision"), Push("Debate"), Pop(), Push("Group assignment")` "Debate" is popped.

Stack now: **["CBE notes", "Math revision", "Group assignment"]**
➤ **Top of stack: "Group assignment"**

Q6: Which answers remain in the stack after undoing?

Let's assume the stack had 5 actions, and 3 were undone (3 pops).
Only the first **2 actions remain** (the bottom two).
➤ The remaining items are the **earliest actions**, since the last 3 were popped.

D. Advanced Thinking

Q7: How does a stack enable this retracing process?

Each step in the RwandAir booking is **pushed** onto a stack. As the user clicks "back," the last step is **popped**, taking them to the previous step. This stack behavior allows step-by-step retracing in reverse order.

Q8: Show how a stack algorithm reverses the proverb “Umwana ni umutware”.

Steps:

Push words: Push("Umwana"), Push("ni"), Push("umutware"

Pop words: "umutware", "ni", "Umwana"

➢ **Reversed: "umutware ni Umwana"**

Q9: Why does a stack suit this case better than a queue?

In DFS (Deep Search), we go deep into one branch before backtracking.
A stack lets us **track the path taken**, backtracking by **popping** previous steps.
A queue (FIFO) would force us to search level by level, not deeply.

Q10: Suggest a feature using stacks for transaction navigation.

Feature: A "Back and Forward" navigation like browser history.

Push each viewed transaction onto a stack.

Going back **pops** from the current stack.

Another stack can store "forward" history for redo functionality.

➢ Part II - QUEUE

A. Basics

Q1: How does this show FIFO behavior?

At a restaurant, the **first customer** to arrive is the **first served**.
Like a queue: items are **enqueued at the rear** and **dequeued from the front**.

Q2: Why is this like a dequeue operation?

In a YouTube playlist, the **first video added** is the **first to play**.
Just like dequeue: remove the front (oldest) item for processing.

B. Application

Q3: How is this a real-life queue?

At RRA offices, people join a line and are served in **order of arrival**.
This mirrors a queue: **Enqueue** as they arrive, **Dequeue** as they're served.

Q4: How do queues improve customer service?

Queues ensure fairness and **orderly service**.
They avoid skipping or confusion, letting staff handle requests **first-come, first-served**.

C. Logical

Q5: Who is at the front now?

Operations:
Enqueue("Alice"), Enqueue("Eric"), Enqueue("Chantal"), Dequeue(), Enqueue("Jean")

○

"Alice" is dequeued.
➢ Queue: **["Eric", "Chantal", "Jean"]**

➢ **Front: "Eric"** ○

Q6: Explain how a queue ensures fairness.

A queue serves people in the order they arrived—**FIFO**.
Everyone waits their turn, no skipping, making it **fair and predictable**.

D. Advanced Thinking

Q7: Explain how each maps to real Rwandan life.

Linear Queue: Guests lining up for food at a wedding. First come, first served.

Circular Queue: Buses at Nyabugogo terminal rotating routes in a loop

Deque: Passengers boarding a bus from either front or back doors.

Q8: How can queues model this process?

Customers place food orders (**enqueue**), and when ready, the restaurant calls them (**dequeue**).
The order queue ensures meals are prepared and served in the right sequence.

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

At CHUK, **emergency patients are treated first**, even if others arrived earlier. This is a **priority queue**, where some items (cases) are handled **before others**, based on importance—not just arrival time.

Q10: How would queues fairly match drivers and students

As passengers request rides, they are **enqueued**.
Drivers are also queued up.
The app matches the **first available driver** with the **first waiting rider**, ensuring fairness in order an