BIT YEAR 2 DATABASE STRUCTURE AND ALGORITHM

PART I: STACK

A. Basics

Q1: How does this show the LIFO nature of stacks?
When I'm using the MTN MOMO app and filling in payment steps, each step I take is like pushing something onto a stack. When I press back, it removes the last step I did — just like popping from a stack. The last thing I did is the first one to go, which clearly shows the LIFO (Last In, First Out) behavior.

Q2: Why is this action similar to popping from a stack?
In UR Canvas, going back through course modules feels like popping from a stack because it removes the most recent page or step I took. It's like undoing what I just did, one by one, from the top — the same way we pop items off a stack.

B. Application

Q3: How could a stack enable the undo function when correcting mistakes?
If every action I take is pushed onto a stack, then when I make a mistake, I can just pop the last few actions to undo them. This makes it easy to go back step-by-step in reverse order, just like how I might undo typing in a text editor.

Q4: How can stacks ensure forms are correctly balanced?
When filling out Irembo forms, if I use a stack to track every section or input field I open, I can make sure I close or complete each one correctly. Just like matching parentheses, I push when I open something and pop when I complete or close it. If the stack is empty at the end, then I know everything is properly matched.

C. Logical

Q5: Which task is next (top of stack)?
Here's what happened:

- Push "CBE notes"
- Push "Math revision"
- Push "Debate"
- Pop (removes "Debate")

- Push “Group assignment”
  So the top of the stack is **“Group assignment”**, because it was the last thing I added after removing “Debate”.

Q6: Which answers remain in the stack after undoing?
If I undo 3 recent actions using pop, that means the top 3 actions are removed. So whatever was below those 3 is what remains. If I had, say, done 5 actions, the bottom 2 are still in the stack. These remaining ones are the earlier ones I did.

D. Advanced Thinking

Q7: How does a stack enable this retracing process?
When booking on RwandAir, each step (like selecting date, destination, seat, etc.) gets pushed onto a stack. If I want to go back, I just pop the last step and return to the previous one. It lets me retrace step-by-step in the reverse order I followed.

Q8: Show how a stack algorithm reverses the proverb “Umwana ni umutware.”
First, I push each word onto the stack:

- Push(“Umwana”)
- Push(“ni”)
- Push(“umutware”)

Then, I pop each word:

- Pop → “umutware”
- Pop → “ni”
- Pop → “Umwana”

So the reversed proverb is: **“umutware ni Umwana.”**

Q9: Why does a stack suit this case better than a queue?
In a deep search at Kigali Public Library, a stack is better because I go deep into one shelf, and if I don’t find the book, I backtrack — just like popping from a stack. A queue would go level by level, which is not efficient when I want to go deep first.

Q10: Suggest a feature using stacks for transaction navigation.
In the BK Mobile app, a stack could be used to let users go back through recent transactions. Every time I view a transaction, it’s pushed onto the stack. If I press “back,” it pops and takes me to the one I saw before — easy for navigation history.

PART II: QUEUE

A. Basics

Q1: How does this show FIFO behavior?
At a restaurant, the first person who comes is the first to be served. That's exactly how a queue works — First In, First Out (FIFO). Everyone waits their turn, and no one skips the line.

Q2: Why is this like a dequeue operation?
In a YouTube playlist, the video that was added first plays first. When it finishes, it gets removed automatically. That's the same as dequeue — removing the front item.

B. Application

Q3: How is this a real-life queue?
At RRA, people waiting to pay taxes stand in line. The first person to join the line is the first to be served. That's a perfect example of a queue in real life — things are processed in the order they arrive.

Q4: How do queues improve customer service?
Queues help keep things fair and organized. Everyone knows their position, and no one gets skipped. This makes service faster and less stressful for both staff and customers.

C. Logical

Q5: Who is at the front now?
Here's what happened:

- Enqueue("Alice")
- Enqueue("Eric")
- Enqueue("Chantal")
- Dequeue() → removes "Alice"
- Enqueue("Jean")

So now the queue looks like: **Eric, Chantal, Jean**
Eric is at the front.

Q6: Explain how a queue ensures fairness.
A queue makes sure that the first person to submit their application is the first to be processed. Everyone waits their turn — no jumping ahead. That's why it's fair.

D. Advanced Thinking

Q7: Explain how each maps to real Rwandan life.

- **Linear queue**: Like people waiting to serve themselves at a wedding buffet — they join at the back and move forward one by one.
- **Circular queue**: Like buses in Nyabugogo — after completing a loop, they go back to the start and wait for the next trip.
- **Deque**: Like boarding a bus from either the front or back door — people can enter or exit from both ends.

Q8: How can queues model this process?
In a restaurant, when I place an order, it gets added to the queue. The kitchen processes them one by one in the order they came in. When food is ready, the order is dequeued and served.

Q9: Why is this a priority queue, not a normal queue?
At CHUK, if someone comes in with a life-threatening emergency, they get treated first, even if they arrived after others. This is a **priority queue** because it doesn't only follow order — it looks at urgency.

Q10: How would queues fairly match drivers and students?
In a moto app, drivers and students can be added to two separate queues. The app pairs the first driver with the first student in line — ensuring no one is skipped and everyone gets a turn in order.