

UNIVERSITY OF RWANDA

COLLEGE: BUSINESS AND ECONOMICS

**DEPARTMENT: BUSINESS INFORMATION TECHNOLOGY**

**REG: 224008051**

**NAME: MBARUSHAMUNGU OLIVIER Date:24/sep/2025**

**Part I – STACK**

**A. Basics**

**Q1:** How does MoMo "back" show LIFO?  
. Because the **last step you entered** (e.g., payment detail) is the **first step removed** when you press back.

**Q2:** Why is Canvas "back" like popping?  
. Pressing back **removes the most recent page/action** just like **Pop removes the top item** in a stack.

**B. Application**

**Q3:** How can stack enable undo?  
. Each action is **pushed** onto the stack. When you undo, you **pop the last action** to return to the previous state.

**Q4:** How can stacks ensure forms are balanced?  
. Push every opening field/bracket; when a matching closing field appears, **pop it**. If all match correctly, the form is balanced.

**C. Logical**

**Q5:** Task sequence: Push(“CBE notes”), Push(“Math revision”), Push(“Debate”), Pop(), Push(“Group assignment”).  
. Stack now = [CBE notes, Math revision, Group assignment].  
. **Top = Group assignment.**

**Q6:** Undo 3 recent actions.  
. Popping 3 times removes the last 3 answers.  
. The **earlier answers remain** at the bottom of the stack.

**D. Advanced Thinking**

**Q7:** How does a stack enable retracing in RwandAir booking?  
. Each booking step is **pushed**; going back **pops** the last step to retrace correctly.

**Q8:** Reverse “Umwana ni umutware” using stack.  
. Push words: [Umwana, ni, umutware].  
. Pop → “umutware ni Umwana”.

**Q9:** Why is stack better than queue in DFS?  
.DFS goes **deep first**, always taking the **last unexplored path**. A stack naturally supports this LIFO behavior.

**Q10:** Suggest a stack feature for BK navigation.  
A **“back” and “forward” button**:

* Back = Pop the last transaction.
* Forward = Push it again when revisiting.

**Part II – QUEUE**

**A. Basics**

**Q1:** Restaurant order shows FIFO?  
.The **first customer seated** is the **first one served** → just like FIFO.

**Q2:** Why is playlist like dequeue?  
.The **next video in front** plays automatically, just like **dequeue removes from the front**.

**B. Application**

**Q3:** RRA tax line = real-life queue?  
.People arrive, **enqueue at the back**; the first person in line **dequeues first** to pay.

**Q4:** How do queues improve service?  
.They keep customers in **order of arrival**, avoiding confusion and ensuring **fairness**.

**C. Logical**

**Q5:** Equity Bank sequence: Enqueue(Alice), Enqueue(Eric), Enqueue(Chantal), Dequeue(), Enqueue(Jean).  
.Queue after steps = [Eric, Chantal, Jean].  
. **Front = Eric.**

**Q6:** How queue ensures fairness in RSSB?  
. Applications are **processed in arrival order** (FIFO), so no one skips the line.

**D. Advanced Thinking**

**Q7:** Real-life queue types:

* **Linear queue:** People line up at a buffet → first in line served first.
* **Circular queue:** Buses at Nyabugogo loop back after dropping passengers.
* **Deque:** Passengers boarding a bus from **front or rear**.

**Q8:** Restaurant orders.  
.Orders are **enqueued when made**; food is prepared; when ready, the order is **dequeued** and served.

**Q9:** Why is CHUK emergencies a priority queue?  
.Patients with emergencies are **served before others**, even if they arrived later.

**Q10:** Moto/E-bike app matching.  
. Drivers are **enqueued when available**, students are **enqueued when requesting**.  
. Matching happens FIFO: **first driver ↔ first student**, ensuring fairness.

REFFERENCE

**[.https://www.lkouniv.ac.in › site › siteContent › 2...](https://www.lkouniv.ac.in/site/writereaddata/siteContent/202003251324427324himanshu_Stack_Queue.pdf)**

**.** **"Data Structures and Algorithms Made Easy" by Narasimha Karumanchi:**

**."Algorithms" by Robert Sedgewick and Kevin Wayne**

**."Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (CLRS):**

**."Data Structures and Algorithms in Java" / "Data Structures and Algorithms in C++" / "Data Structures and Algorithms in Python" by Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser**