

**UNIVERSITY OF RWANDA**

**SCHOLL OF BUSINESS**

**COLLEGE OF BUSINESS AND ECONOMIC**

**DEPARTMENT OF BUSINESS INFORMATION AND TECHNOLOGY**

**FIRST NAME:diogene**

**SURNAME:NKUNDIMANA**

**REG NUMBER:224013914**

**BIT – Data Structure Exercise – Number 2**

**Part I – STACK**

**Stack:**is linear structure that store element in sequences

**Push :** is operation of inserting or adding element into top of stack

**Pop:** is operation of removing or deleting the top element from stack

**Lifo(last in first out) :**is principle that govern stack operation

**Example:**inserting 1,2 and removing3

**Fifo (first in first out):** rule s first element that you put in first one you take

**Q1: How does this show the LIFO nature of stacks?**  
In the MTN MoMo application, when a user is filling out payment details, each step is completed one after another, like entering the recipient’s number, amount, and then confirming the PIN. These steps are stored in memory in the order they are added

**Q2: Why is this action similar to popping from a stack?**  
When navigating course modules in UR Canvas, pressing "back" undoes the most recent step. For example, if you open Module 1, then Module 2, and then Module 3, the system keeps a stack of your navigation steps. When you press back, the system pops the last module visited (Module 3) and takes you back to Module 2. This is the same as popping in a stack, where the most recently added item (on top) is the first to be removed.

**B. Application**

**Q3: How could a stack enable the undo function when correcting mistakes?**  
In BK Mobile Banking, each transaction (deposit, withdrawal, transfer) can be pushed onto a stack representing transaction history. If a user makes a mistake, the undo function works by popping the last action performed. This allows the app to remove the most recent transaction while keeping earlier ones intact.

**Q4: How can stacks ensure forms are correctly balanced?**  
When filling forms like those in Irembo registration, data entry fields often need to be correctly matched — for example, ensuring every opening bracket has a corresponding closing bracket. Using a stack, every time an opening bracket or tag is entered, it is pushed onto the stack. When a closing bracket appears, the system checks the top of the stack for a match and pops it if correct. If the stack is empty at the right times and everything matches, the form is balanced. This prevents errors like unclosed fields or unmatched data.

**C. Logical**

**Q5: Which task is next (top of stack)?**  
The sequence is:

* Push (“CBE notes”) → Stack: [CBE notes]
* Push (“Math revision”) → Stack: [CBE notes, Math revision]
* Push(“Debate”) → Stack: [CBE notes, Math revision, Debate]
* Pop () → Stack: [CBE notes, Math revision]
* Push (“Group assignment”) → Stack: [CBE notes, Math revision, Group assignment]

the top of the stack (the next task) is **“Group assignment.”**

**Q6: Which answers remain in the stack after undoing?**  
If a student undoes 3 recent actions, the system performs three pops on the stack. That means the last three actions are removed, leaving only the earlier ones still stored. In this way, only the initial answers (before the last three changes) remain in the stack. The stack helps maintain what is left, while discarding the latest operations.

**D. Advanced Thinking**

**Q7: How does a stack enable this retracing process?**  
When booking with RwandAir, a passenger might enter details like flight selection, passenger details, payment information, and confirmation. These steps are stored on a stack. If the user decides to go back step-by-step, the system pops the last step (confirmation), then the payment information, then the passenger details. This retracing is possible because the most recent step is always on top, making stacks ideal for backtracking through processes.

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

* Push(“Umwana”) → Stack: [Umwana]
* Push) → Stack:Umwana, ni
* Push(“umutware”→ Stack: Umwana, ni, umutware

Now, popping each word reverses the order:

* Pop → “umutware
* Pop →ni
* Pop → “Umwana”

Reversed result: **“umutware ni Umwana.”**

**Q9: Why does a stack suit this case better than a queue?**  
In a library deep search (DFS – Depth-First Search), the system explores as far as possible down one path before backtracking. A stack naturally supports this because the most recent node visited is stored on top, and backtracking simply involves popping that node. A queue (FIFO) is better suited for breadth-first search (BFS), which explores level by level. But for DFS, where retracing is necessary, the stack is the perfect tool.

**Q10: Suggest a feature using stacks for transaction navigation.**  
In BK Mobile app, stacks could be used to allow users to move backward and forward through their transaction history. Each time a user opens a transaction, it is pushed onto the stack. If they press “back,” the most recent one is popped, showing the previous one. A “redo” feature could also be implemented using another stack to temporarily hold popped items, allowing users to return forward if needed.

**Part II – QUEUE**

**A. Basics**

**Q1: How does this show FIFO behavior?**  
At a Kigali restaurant, customers are served in the exact order they arrive. The first customer to enter is the first to be served, while later customers must wait their turn. This reflects *First-In, First-Out (FIFO)*, the main property of queues: the earliest arrivals leave the queue first.

**Q2: Why is this like a dequeue operation?**  
In a YouTube playlist, videos are arranged in a line. The first video added to the queue plays first. Once it finishes, it is dequeued (removed from the front), and the next one plays. This automatic removal from the front of the line makes it similar to the dequeue operation.

**B. Application**

**Q3: How is this a real-life queue?**  
At RRA offices, taxpayers arrive and stand in line to pay. The person who comes first is served first, and those who come later wait behind. This exactly matches the structure of a queue where elements are enqueued at the rear and dequeued from the front.

**Q4: How do queues improve customer service?**  
In service centers like MTN or Airtel, queues ensure that requests such as SIM replacements are handled fairly and in order of arrival. Without queues, customers might argue or feel neglected. With a queue system, everyone knows their position, and no one can jump ahead unfairly. This reduces stress, increases fairness, and helps staff work systematically.

**C. Logical**

**Q5: Who is at the front now?**  
The sequence is:

* Enqueue(“Alice”) → Queue: [Alice]
* Enqueue(“Eric”) → Queue: [Alice, Eric]
* Enqueue(“Chantal”) → Queue: [Alice, Eric, Chantal]
* Dequeue() → Queue: [Eric, Chantal] (Alice leaves)
* Enqueue(“Jean”) → Queue: [Eric, Chantal, Jean]

So, the person at the front is **Eric.**

**Q6: Explain how a queue ensures fairness.**  
In systems like RSSB pension applications, each applicant is served in the exact order they arrived. A queue structure prevents anyone from skipping ahead. This ensures fairness because the first application submitted is the first one processed, and later submissions must wait their turn. The FIFO system avoids confusion or unfair delays.

**D. Advanced Thinking**

**Q7: Explain how each maps to real Rwandan life.**

* **Linear queue:** At a wedding buffet, people join the line one after another, and each takes food in order. The first to join the line is the first to be served.
* **Circular queue:** At Nyabugogo bus station, buses keep looping routes. After reaching the end, they return to the start, just like circular queues wrap around.
* **Deque:** When boarding buses, passengers sometimes enter from the front or the rear. A double-ended queue allows adding or removing items from both ends, just like boarding from either side.

**Q8: How can queues model this process?**  
In a Kigali restaurant, customers place orders (enqueue), and the chef prepares them in order. When an order is ready, the customer is called (dequeue). This ensures that the first order placed is the first one served, creating fairness and efficiency.

**Q9: Why is this a priority queue, not a normal queue?**  
At CHUK hospital, emergencies are treated before normal patients, regardless of arrival time. In a standard queue, the first patient to arrive should be served first. But in a priority queue, the most urgent cases are given priority, meaning the order is determined by priority, not just arrival.

**Q10: How would queues fairly match drivers and students?**  
In a moto/e-bike taxi app, students requesting rides are placed in a queue. Drivers waiting for passengers are also in a queue. The system dequeues one driver and one student at a time, matching them fairly. This ensures both students and drivers are served in the order they arrived, preventing unfairness and long waits.