UNIVERSITY of RWANDA

SCHOOL OF BUSINESS

COLLEGE OF BUSINESS AND ECONOMICS

DEPARTMENT OF BUSINESS INFORMATION TECHNOLOGY

LECTURER: DR RUKUNDO PRINCE

NAMES	*REG NUMBER*
IRADUKUNDA CLEMENTINE	*224010270*

DATA STRUCTURES AND ALGORITHMS, BIT80331

QUESTIONS ON STACKS AN

9/24/2025

Part I - STACK

A. Basics

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

Answer: The MTN MoMo app's payment process involves a sequence of steps (e.g., enter amount, enter PIN, confirm). Each completed step is "pushed" onto a navigation stack. When you press "Back," the app "pops" the most recently added step (the last step you completed), returning you to the previous step. This demonstrates Last-In, First-Out (LIFO) because the last step added is the first one removed.

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

Answer: In UR Canvas, navigating course modules involves "pushing" each new page onto a stack. Pressing "Back" "pops" the current page (the top of the stack), revealing the previous page. This is similar to popping from a stack because it removes the most recently added item, undoing the last navigation step.

B. Application

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

Answer: In BK Mobile Banking, each transaction or action (e.g., entering text, selecting an option) is recorded as an "action object" and "pushed" onto a stack. When a user corrects a mistake by triggering "Undo," the application "pops" the most recent action from the stack and reverses it. This allows mistakes to be undone in the reverse order they were made, following LIFO.

Q4: How can stacks ensure forms are correctly balanced?

Answer: In Irembo registration forms, fields may require balanced symbols (e.g., parentheses, brackets). A stack can check this by:

- Iterating through each character in the form data.

- Pushing

opening symbols (e.g., `(`, `[`, `{`) onto the stack.

- When a closing symbol (e.g., `)`, `]`, `}`) is found,

-popping

the top of the stack and checking if it matches the closing symbol.

- If the stack is empty at the end and all symbols match, the form is balanced; otherwise, it indicates an error.

C. Logical

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

Answer: The sequence of operations 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 is "Group assignment", so it is the next task.

Q6: Which answers remain in the stack after undoing?

Answer: Assume the student had answered 5 questions in order, with answers pushed onto a stack: `[Ans1, Ans2, Ans3, Ans4, Ans5]` (Ans5 is the top). Undoing 3 actions means popping three times: Pop() removes Ans5, Pop() removes Ans4, Pop() removes Ans3. The stack now contains `[Ans1, Ans2]`. Thus, Ans1 and Ans2 remain.

D. Advanced Thinking

Q7: How does a stack enable this retracing process?

- **Answer:** In RwandAir booking, each step (e.g., select flight, enter passenger details, choose seats) is "pushed" onto a navigation stack as the passenger completes it. Pressing "Back" "pops" the current step from the stack, returning the passenger to the previous step. The stack stores the history of steps, allowing step-by-step backtracking in LIFO order.

Q8: Show how a stack algorithm reverses the proverb.

Answer: The proverb "Umwana ni umutware" is split into words: "Umwana", "ni", "umutware".

Push

each word onto the stack:

- Push("Umwana") → Stack: `[Umwana]`
- Push("ni") → Stack: `[Umwana, ni]`
- Push("umutware") → Stack: `[Umwana, ni, umutware]`
- **Pop**

each word from the stack (LIFO order):

- Pop() → "umutware"

- Pop() → "ni"

- Pop() → "Umwana"

- The reversed proverb is "umutware ni Umwana".

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

-**Answer:**A Depth-First Search (DFS) explores one path as deeply as possible before backtracking. A stack is ideal because:

- When a student discovers a new shelf (node), it is pushed onto the stack.

- To choose the next shelf, the student pops the most recently discovered shelf (top of the stack), encouraging deep exploration.

- A queue would implement Breadth-First Search (BFS), where shelves are explored level by level (FIFO), which is less efficient for a deep search.

Q10: Suggest a feature using stacks for transaction navigation.

-**Answer**: A "Transaction Drill-Down" feature in the BK Mobile app:

- The main transaction list is the base level (Stack Level 1).

- Selecting a transaction pushes a detailed view onto the stack (Level 2).

- Clicking "View Receipt" pushesthe receipt onto the stack (Level 3).

- The "Back" button pops the current view, returning to the previous level. This provides intuitive, LIFO-based navigation.

Part II - QUEUE

A. Basics

Q1: How does this show FIFO behavior?

-**Answer:** At a restaurant, customers enqueuer by joining the line at the rear. The chef dequeues by serving the customer at the front first. The first customer to arrive is the first to be served (First-In, First-Out), demonstrating FIFO.

Q2: Why is this like a dequeue operation?

- **Answer:**A YouTube playlist is a queue of videos. When a video ends, it is dequeued (removed from the front of the queue), and the next video (which was added after the current one) automatically plays. This is similar to dequeueing because the next item in line is processed.

B. Application

Q3: How is this a real-life queue?

-**Answer:**At RRA offices, people Nenqueue by taking a ticket number or forming a physical line, representing their position in a queue. Tax officers dequeue by calling the next number or serving the person at the front, processing requests in arrival order.

Q4: How do queues improve customer service?

-**Answer:** Queues ensure fairness by processing requests in FIFO order, preventing chaos. Customers know their wait time is based on arrival order, and service centers can manage resources efficiently by handling requests sequentially.

C. Logical

Q5: Who is at the front now?

- **Answer:**The sequence of operations is:

  - Enqueue("Alice") → Queue: `[Alice]` (front: Alice)

  - Enqueue("Eric") → Queue: `[Alice, Eric]` (front: Alice)

  - Enqueue("Chantal") → Queue: `[Alice, Eric, Chantal]` (front: Alice)

  - Dequeue() → Queue: `[Eric, Chantal]` (front: Eric)

  - Enqueue("Jean") → Queue: `[Eric, Chantal, Jean]` (front: Eric)

- **Eric** is at the front of the queue.

Q6: Explain how a queue ensures fairness.

-**Answer:** RSSB pension applications are enqueued in the order they are received. The processing system dequeues applications from the front, always handling the oldest application first. This FIFO discipline ensures that no application is skipped or delayed unfairly, guaranteeing fairness based on arrival time.

D. Advanced Thinking

Q7: Explain how each maps to real Rwandan life.

Linear Queue: Like people at a wedding buffet—each person joins at the rear and is served from the front, moving forward sequentially. The queue has a clear start and end.

Circular Queue: Like buses at Nyabugogo taxi park—a fixed number of buses cycle on a route. When a bus departs (dequeued), it returns to the end of the line after completing its trip, reusing resources efficiently.

-**Deque** (Double-Ended Queue): Like boarding a bus—people can enter from the rear (normal enqueue) or sometimes from the front if space allows, and can exit from either end, providing flexibility.

Q8: How can queues model this process?

- **Answer:** At a Kigali restaurant, two queues can model the process:

- **Order Queue:** When a customer orders, the order is enqueued into a kitchen queue. The chef dequeues orders from the front to prepare them.

- **Ready Queue:** When food is ready, the order is enqueued into a "ready" queue. Servers dequeue from the front to call customers.

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

-**Answer:** At CHUK hospital, patients are served based on medical urgency (priority), not arrival time. A priority queue dequeues the patient with the highest priority first (e.g., emergencies), while a normal queue would strictly follow FIFO. This allows critical cases to "jump the line."

Q10: How would queues fairly match drivers and students?

- **Answer:** In a moto/e-bike taxi app, two FIFO queues are used:

- **A driver queue** where available drivers enqueue.

- **A student/rider queue** where waiting students enqueue.

- The system dequeues the driver who has waited longest and the student who has waited longest, matching them fairly. This ensures first-come, first-served fairness for both parties.