NAMES: UMUTONIWASE MARIE JUSTINE

REG NUMBER:224008785

GROUP 1 BIT

DATA STRUCTURE ASSIGNMEN

Part1: stack

A stack is a data structure that works on the principle of LIFO (Last in First Out). The latest element that is inserted is always the first to be removed. stacks are used in many real-life applications such as undo operations, navigation, expression evaluation and backtracking.

A. BASICS

1.Push/POP (LIFO) in a stack the last item added is the first removed. In the MTN MOMO app, when you fill payment details step-by step, pressing back removes the last step

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

It shows LIFO because the last stack added(debate) was the first one removed by the first Pop () operation. The first task added (CBE notes) remains at the bottom until the very end.

2.**why is pressing "back'' similar to popping from a stack?**

Each steps in a process (like filling a form or navigating pages) is "pushed" into a navigation stack. Pressing "back" reverses the most recent step, which is exactly what pop () does it removes and returns to the previous state. Because when you undo multiple steps, you remove the most recent actions first, just like popping from the top of a stack.

A. Application

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

Each action is pushed onto the stack. If a mistake is made, the system can pop recent actions to undo them in reverse order, restoring the previous state.

Q4: **How can stacks ensure forms are correctly balanced?**

By pushing opening brackets/fields onto the stack and popping when a matching closing bracket/field is found. If the stack is empty at the end and no mismatch occurs, the form is balanced.

B. Logic

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

Given sequence:

Push (“CBE notes”): stack = [CBE notes]

Push (“Math revision”): stack = [CBE notes, Math revision]

Push(“Debate”): stack = [CBE notes, Math revision, Debate]

Pop (): removes “Debate” → stack = [CBE notes, Math revision]

Top of stack (next task) = “Math revision”

Q6: **Which answers remain in the stack after undoing?**

Continuing from above:

Current stack = [CBE notes, Math revision]

Then: Push (“Group assignment”): stack = [CBE notes, Math revision, Group assignment]

After undo (Pop):

Pop () removes “Group assignment” : stack = [CBE notes, Math revision]

Remaining in stack: “CBE notes”, “Math revision”

Q7: How does a stack enable this retracing process?

A stack works in LIFO (Last in, First Out) order. When retracing (like going back in Rwanda Air booking or transaction history), the last step taken is the first to be undone. This makes a stack perfect for step-by-step backtracking.

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

1. Push each word onto the stack:

Push(“Umwana”)

Push(“ni”)

Push(“umutware”)

2. Pop each word from the stack to build the reversed sentence:

Pop: “umutware”

Pop: “ni”

Pop: “Umwana”

Reversed proverb: “umutware ni Umwana.”

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

- Because in navigation or backtracking, you always need to access the most recent step first.

  Stack (LIFO): Perfect for undo and backtracking.

  Queue (FIFO): Works in first-come, first-served order, which is not useful for retracing.

  So, a stack suits better since it naturally follows the order needed for reversing or going back.

  Q10**: Suggest a feature using stacks for transaction navigation.**

A "Back" and "Forward" feature in BK Mobile app:

Back button: Pops the last viewed transaction (go to the previous one).

Forward button: Uses another stack to redo the step (similar to a browser's back/forward navigation).

This allows the user to move smoothly through transaction history without losing their place

PART 2: QUEUE

A. Basic

1: How does this show FIFO behavior?

- Because in a queue, the First-In is the First-Out. At a restaurant in Kigali or RRA offices, the first customer who joins the line will be the first one served, while others wait their turn in order.

Q2: **Why is this like a dequeue operation?**

A dequeue means removing the front item of the queue. In MTN/Airtel service centers, the next person waiting is called and served first, just like in a YouTube playlist where the next video (front of the queue) is played automatically.

B.Application

Q3: How is this a real-life queue?

It is a real-life queue because people or tasks are arranged in a line, and they are processed one by one from the front, whether it's customers waiting at RRA, restaurant orders, or services at MTN/Airtel.

Q4: How do queues improve customer service?

Queues improve service by making it fair, organized, and efficient. Everyone is served in order, no one is skipped, and confusion is avoided. This reduces waiting conflicts and ensures smooth flow of operations in restaurants, offices, or service centers.

C. Logical

Q5: Who is at the front now?

After Enqueue("Alice"), Enqueue("Eric"), Enqueue("Chantal"), Dequeue()

Alice was first, but she is removed by Dequeue().

Now Eric is at the front of the queue.

Q6: Explain how a queue ensures fairness.

A queue follows FIFO (First in, First Out): the first person to arrive is the first to be served.

Example: At RSSB pension offices, people are served in the order they arrived. This avoids skipping and ensures fairness.

D. Advanced thinking

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

Circular queue; Buses looping at Nyabugogo bus park, once they reach the end they restart the route.

Priority queue; At CHUK hospital, emergencies are treated before regular patients.

Linear queue, People lining up at a wedding buffet or waiting in Equity Bank.

Deque, boarding a bus where some passengers enter from the front and others from the rear.

FIFO message handling: In MTN/Airtel SMS systems, the first message sent is the first to be delivered.

Enqueue orders/Dequeue ready: At a Kigali restaurant, customers order food, and when ready, the first order is served first.

Enqueue/Dequeue matching system: In a moto/e-bike taxi app, riders (enqueue) wait until passengers (dequeue) match with them.

Q8: How can queues model this process?

Queues model real-world waiting processes because they represent fairness and order:

Each new request/person is added to the rear.

The one waiting longest at the front is served next.

This is seen in banks, hospitals, restaurants, and transport apps.

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

Because in a priority queue, some items jump ahead based on urgency.

Example: At CHUK, an accident victim is treated before someone who came earlier with a mild headache. In a normal queue, order is strictly by arrival time, but here, importance/urgency decides.

Q10: How would queues fairly match drivers and students?

By using a matching system (enqueue & dequeue):

Students (enqueue) request rides.

Drivers (enqueue) offer seats.

The system matches the first student waiting with the first available driver, ensuring fairness (FIFO). This avoids skipping and guarantees everyone is served in the order they arrived.