NAME:MUGISHA   EDWARD

REG:224006087

DATE:23 sept 2025

BIT - Data structure and algorithm exercise

A. **Basics**

1. Operation: 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?

When you're filling out payment details on the MTN MoMo app, each step you complete is like pushing a new item onto a stack. The "back" button then acts like a pop operation, removing the most recently added step. This perfectly demonstrates the Last-In, First-Out (LIFO) principle because the last step you took (the one you just completed) is the first one to be re n moved when you press "back."

2. Operation: Pop (Undo) — Pop removes the top item. In UR Canvas, when you navigate course modules, pressing back undoes the last step.

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

The "back" button on UR Canvas, like many navigation systems, allows you to undo your last action, which in this case is navigating to a new module. Each module you visit is essentially added to a temporary history stack. When you press "back," you're removing the most recent page (or the "top item") from this stack and returning to the previous one. This is exactly how the pop operation works on a stack: it removes the top, or last-added, item. This behavior is a very common real-world application of stacks.

B. Application

3. Operation: Push (Add to stack) — New actions are added to the stack top. In BK Mobile Banking, transactions are added to history.

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

A stack can enable an undo function by storing a history of actions. When a user performs an action (e.g., typing text, applying a filter, or a transaction), that action is pushed onto a stack. When the user wants to undo a mistake, the application performs a pop operation, removing the most recent action from the stack and reverting the state of the application to what it was before that action was performed. Each time the user hits "undo," the application pops the next item off the stack, effectively unwinding the action history in reverse chronological order.

4. Operation: Balanced Parentheses Check (Stack-based matching) — Push opening bracket, pop when matching closing bracket is found. In Irembo registration forms, data entry fields must be correctly matched.

Q4: How can stacks ensure forms are correctly balanced?

A stack can check if a form is balanced by pushing each opening field onto the stack. When a closing field is found, it pops an item from the stack. The form is balanced only if the stack is empty at the end. If not, there's a missing match.

C.Logical

5. Operation: Push and Pop sequence. A student records tasks in a stack: Push (“CBE notes”), Push (“Math revision”), Push(“Debate”), Pop (), Push (“Group assignment”)

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

Push ("CBE notes"): The stack now has [CBE notes].

Push ("Math revision"): The stack becomes [CBE notes, Math revision].

Push("Debate"): The stack is now [CBE notes, Math revision, Debate].

Pop (): The last item added, "Debate," is removed. The stack is now [CBE notes, Math revision].

Push ("Group assignment"): The new task is added to the top. The stack is now [CBE notes, Math revision, Group assignment].

The next task is "Group assignment".

6. Operation: Undo with multiple Pops. During ICT exams, a student undoes 3 recent actions.

Q6: Which answers remain in the stack after undoing?

null

D. Advanced Thinking

7. Operation: Pop to backtrack. In Rwanda Air booking, a passenger goes back step-by-step in the form.

Q7: How does a stack enable this retracing process?

When a user fills out a form step-by-step, each step is pushed onto a stack. To retrace, the "back" button performs a pop operation, removing the most recent step and taking the user back to the previous one. This is a classic LIFO (Last-In, First-Out) application.

8. Operation: Push words, then Pop to reverse. To reverse “Umwana ni umutware”, push each word and then pop.

Q8: Show how a stack algorithm reverses the proverb.

To reverse "Umwana ni umutware" using a stack, you would:

1. Push "Umwana"
2. Push "ni"

3. Push "umutware"

The stack now has [Umwana, ni, umutware]. Then, pop each word one by one. The output would be:

- Pop: "umutware"
- Pop: "ni"
- Pop: "Umwana"

9. Operation: DFS using a stack. A student searches shelves in Kigali Public Library (deep search).

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

A stack is better for a deep search (like a DFS) because it prioritizes going as deep as possible into one path before exploring others. When a new shelf (or node) is found, it's pushed onto the stack. The search then continues from that new location. A queue would search "level by level," which is not what a deep search does. The LIFO nature of a stack naturally supports the "go deep, then backtrack" behavior of a DFS.

10. Operation: Push/Pop for navigation. In BK Mobile app, moving through transaction history uses push and pop.

Q10: Suggest a feature using stacks for transaction navigation.

A great feature would be a "Go Back to Previous Transaction" button. This would allow users to navigate back to a specific transaction they viewed earlier. Every time a user clicks on a transaction, its ID would be pushed onto a stack. The "Go Back" button would then perform a pop to retrieve and display the previous transaction, acting like a quick, in-app browser history**.**

-----------------------Part II- QUEUE

A. Basics

1. Operation: Enqueue (add at rear), Dequeue (remove from front). At a restaurant in Kigali, customers are served in order.

Q1: How does this show FIFO behavior?

The first customer in line is the first one to be served, and the next person in line is served next. This is exactly how a FIFO works

2. Operation: Dequeue (next item leaves first). In a YouTube playlist, the next video plays automatically.

Q2: Why is this like a dequeue operation?

Playing the next video in a YouTube playlist is like a dequeue operation because the video that has been waiting the longest (at the front of the "queue") is the one that plays next. Once it finishes, it's removed from the front of the list, and the next video automatically starts.

B. Application

3. Operation: Enqueue (job submission). At RRA offices, people waiting to pay taxes form a line.

Q3: How is this a real-life queue?

This is a real-life queue because the people waiting in line are served in the order they arrived. New people join the end of the line (enqueue), and the person at the front of the line is served and leaves (dequeue). This perfectly demonstrates the FIFO (First-In, First-Out) principle.

4.Operation: Queue management. In MTN/Airtel service centers, SIM replacement requests are processed in order.

Q4: How do queues improve customer service?

Queues improve customer service by creating a fair and orderly system. By processing requests in the order they were received, a queue ensures that no one is served out of turn. This prevents confusion, reduces wait-time uncertainty, and gives customers confidence that their request will be handled fairly and efficiently.

C. Logical

5.Operation: Sequence of Enqueue/Dequeue. In Equity Bank, operations are: Enqueue(“Alice”), Enqueue(“Eric”), Enqueue(“Chantal”), Dequeue (), Enqueue(“Jean”)

Q5: Who is at the front now?

1. Enqueue("Alice"): The queue is now [Alice].
2. Enqueue("Eric"): The queue is now [Alice, Eric].
3. Enqueue("Chantal"): The queue is now [Alice, Eric, Chantal].
4. Dequeue (): Alice, who was first in line, is removed. The queue is now [Eric, Chantal].
5. Enqueue("Jean"): Jean is added to the end. The queue is now [Eric, Chantal, Jean].
   After all the operations, Eric is at the front of the queue.

6. Operation: FIFO message handling. RSSB pension applications are handled by arrival order.
   Q6: Explain how a queue ensures fairness.

   A queue ensures fairness by processing items in the exact order they were received. This First-In, First-Out (FIFO) principle prevents any item from being handled out of turn, guaranteeing that every application, request, or customer is served fairly based on their arrival time.

C. Advanced Thinking

7. Operation: Different queue types. Examples: • Linear queue = people at a wedding buffet. • Circular queue = buses looping at Nyabugogo. • Deque = boarding a bus from front/rear.
Q7: Explain how each map to real Rwandan life.

- Linear Queue: People at a wedding buffet form a single line (linear queue). They enter from one end and are served one by one from the other.
- Circular Queue: Buses looping at Nyabugogo bus park are a good example. They form a continuous loop, with buses entering at the front and then moving to the back after a certain time or when a new bus arrives. This allows for efficient use of space.

- Deque: Boarding a bus from the front and back is a Deque (double-ended queue). People can get on or off from either the front or the rear, allowing entry and removal from both ends.

8.Operation: Enqueue orders, Dequeue when ready. At a Kigali restaurant, customers order food and are called when ready.

Q8: How can queues model this process?

This can be modeled using two queues. Customers place an order and are added to a first-come, first-served queue. The orders are then processed by the kitchen. When an order is ready, the customer is removed from the front of the queue and called. This ensures customers are served in the order their food was ready, not just the order they arrived

9. Operation: Priority queue. At CHUK hospital, emergencies jump the line.

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

This is a priority queue because a normal queue serves people in the order they arrived, regardless of their condition. In a hospital, the severity of the emergency is the priority. A patient with a severe injury will "jump the line" and be served before a person with a less serious condition, even if the person with the less serious condition arrived first.

10.Operation: Enqueue/Dequeue matching system. In a moto/e-bike taxi app, riders wait for passengers.

Q10: How would queues fairly match drivers and students?

A queue can match drivers and students fairly by using a FIFO (First-In, First-Out) system. When a student requests a ride, they are placed in a queue. The next available driver is also placed in a queue. The system then matches the student who has been waiting the longest with the driver who has been waiting the longest, ensuring a fair distribution of requests and drivers.

…….END……