DATA STRUCTURE EXERERCIS-NUMBER2

PART I : Stack

Q1: this show the LIFO because the first item added is the first to be removed when we did POP that is the essence of the last in, first out

Q2: This action is similar to the popping from stack because when undoing, the system removes the last action you performed, just like pop removes the most recent item from the stack

Q3: the stack enables the undo function to correcting mistakes when every action is pushed into the stack. To undo, the system simply pops the last actions off stack until you return to the correct state

Q4: Stack ensure forms are correctly balanced when balanced parentheses or matching field by;

- Push when an opening symbol/field is seen
- Pop when closing symbol/field is found.
- If the stack is empty at the end ‘n balanced this ensure data entry fields (like in irembo forms) match correctly

Q5: The next action to top the stack after sequence are;

- Push CBE ‘n [CBE notes]
- Push math revision ‘n [CBE notes math revision]
- Push debate ‘n [CBE notes, math revision
- Pop ‘n removes debate ‘n [CBE NOTES MATH REVISION]
- Push group assignment ‘n [CBE notes, math revision, group assignment]

Top of stack=group assignment

Q6: answers remain in the stack after undoing are

After the above sequence, the stack contains;

- CBE notes bottom
- Math revision
- Group assignment(top)

Q7: Stack retrace each stapes is pushed into the stack when retracing, you pop steps one by one to move backward

Q8: Reverse the program the proverb push each word of “umwana ni umutware” onto the stack, then

pop

- Push: [umwana ni umutware]
- POP(reverse): [umutware ni umwana]

Q9: The stack suits this case better than queue because you need to go back steps by steps(LIFO). A queue would process items in the order they come, which doesn’t allow easy backtracking

Q10: Features using stacks for transaction navigation “back/undo” button in BK mobile that lets users retrace their last steps or undo a transaction entry

<u>PART II- QUEQUE</u>

Q1: this show FIFO behavior in RwandAir booking, the first steps filled is the first one processed (first in first out)

Q2: This is like dequeuer operation because the next passenger in line (front of the queue) is the one served first just like removing from the front in queue

Q3: Real-life queue in Kigali public library, a student searching shelves moves one by one in order, just like items are processed sequentially in a queue

Q4: The queue improves the customer services trough to ensure fairness and order: the first customer to arrive is the first to serve, reducing confusion and keeping service smooth.

Q5: Sequence; Enqueuer(“Alice”), enqueuer (“Eric), enqueuer(“Chantal”), dequeuer (), enqueuer(“jean”)

- After enqueuer ‘n queue= [Alice, Eric, Chantal]
- Dequeuer remove Alice ‘queue= [Eric and Chantal]
- Enqueuer Jean ‘n queue= [Eric, Chantal, Jean]

Q6: Explain how a queue ensures fairness.

A queue works on FIFO (First In, First Out).

The first person to arrive is the first to be served.

This prevents others from “jumping the line” unless it’s a priority queue (e.g., emergencies at CHUK).

✅ Fairness comes from serving people strictly in arrival order.

Q7: Explain how each maps to real Rwandan life.

Linear queue → People lining up at a wedding buffet (first come, first served).

Circular queue → Buses looping at Nyabugogo terminal (when one trip ends, the bus returns to the back to start again).

Priority queue → CHUK hospital emergencies (critical patients are treated first, not by arrival time).

Deque → Boarding a bus where people can enter from both the front or rear door.

Normal queue → RSSB pension applications or bank services (served by order of arrival).

Message queue (FIFO handling) → MTN/Airtel SMS delivery system, where messages are delivered in the same order they were sent.

Q8: How can queues model this process?

In a Kigali restaurant, orders are enqueuer when customers place them and dequeuer when food is ready.

In a moto/e-bike taxi app, riders are enqueuer when they request a trip, and driver's dequeuer them when they accept.

✅ Queues model the waiting and serving process in real time.

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

Because not everyone is served by arrival order.

Example: In CHUK, an emergency patient who arrived later can be served before a regular patient.

✅ That breaks normal FIFO → it's a priority queue.

Q10: How would queues fairly match drivers and students?

Each student request is placed in a queue.

Each driver available also joins a queue.

A matching system dequeuers one student and one driver at the same time → pairing them fairly in order of arrival.

✅ Ensures no driver or student is skipped unless priority rules apply.