BIT - Data Structure Exercise - Number 2

Part I:STACK

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

Answer: In the MTN MoMo app, the steps you follow when entering payment details are stored one after another, just like items being pushed into a stack. When you press the back button, the last step you completed is the first one removed. This demonstrates the Last-In, First-Out (LIFO) principle of stacks, since the most recent entry is undone first.

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

Answer: In UR Canvas, pressing the back button removes the most recent navigation step, just as the pop operation removes the item on top of the stack. The idea is that you can only undo the most recent action, not an earlier one.

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

Answer: When performing tasks such as typing or making transactions, each action is pushed onto a stack. If a mistake occurs, the undo function pops the last action off the stack, restoring the system to the state before the error. This allows users to correct mistakes step-by-step in reverse order.

Q4: How can stacks ensure forms are correctly balanced?

Answer: In forms like Irembo registration, stacks can be used to check for balanced parentheses or matched fields. Each opening bracket or required field is pushed onto the stack, and each closing bracket or matched field pops one item. If the stack is empty at the end, the form is correctly balanced.

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

Answer: The operations are: Push("CBE notes"), Push("Math revision"), Push("Debate"), Pop(), Push("Group assignment"). After these, the stack contains: CBE notes → Math revision → Group assignment. So the next task on top is "Group assignment."

Q6: Which answers remain in the stack after undoing?

Answer: If a student undoes three recent actions, three items are popped from the stack. The remaining stack will contain the earlier answers that were not undone, since popping removes only the most recent items.

Q7: How does a stack enable this retracing process?

Answer: In RwandAir booking, each step is pushed into the stack. When a passenger retraces steps, the pop operation removes the last action, allowing them to move backwards step-by-step until they reach the desired point. This mimics the backtracking process.

Q8: Show how a stack algorithm reverses the proverb.

Answer: To reverse 'Umwana ni umutware', push each word into the stack: [Umwana, ni, umutware]. Popping them one by one gives: 'umutware ni Umwana'. This demonstrates how stacks can reverse sequences.

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

Answer: In depth-first search (DFS), the stack structure is ideal because it explores one branch deeply befo

re backtracking. A queue, by contrast, explores level by level (breadth-first search). Since the student searches deeply into shelves before moving sideways, a stack is better.

Q10: Suggest a feature using stacks for transaction navigation.

Answer: In the BK Mobile app, a stack can be used to store each visited transaction. Users can move forward or backward through history by pushing and popping. This ensures smooth navigation where the most recent viewed transaction is always accessed first.

Part II - QUEUES

Q1: How does this show FIFO behavior?

Answer: In a Kigali restaurant, customers are served in the order they arrive. The first

customer to enter is also the first to be served, while the last customer must wait. This is First-In, First-Out (FIFO) behavior, exactly like a queue.

Q2: Why is this like a dequeue operation?

Answer: In a YouTube playlist, the next video automatically plays by removing it from the front of the list, just like dequeue removes the first element from the queue.

Q3: How is this a real-life queue?

Answer: At RRA offices, people waiting to pay taxes form a line. The person who arrived first is attended first, while latecomers wait their turn. This shows how enqueue and dequeue operations apply to real life.

Q4: How do queues improve customer service?

Answer: In MTN/Airtel service centers, queues ensure customers are helped in the correct order of arrival. This improves fairness and reduces confusion by preventing people from skipping ahead, leading to better customer service.

Q5: Who is at the front now?

Answer: The operations are: Enqueue(Alice), Enqueue(Eric), Enqueue(Chantal), Dequeue(), Enqueue(Jean). After dequeue, Alice is removed. The queue is now [Eric, Chantal, Jean]. So the person at the front is Eric.

Q6: Explain how a queue ensures fairness.

Answer: In RSSB pension applications, the queue ensures requests are handled in the order they arrive. This prevents favoritism and guarantees fairness, since no one can bypass the process.

Q7: Explain how each maps to real Rwandan life.

Answer: Linear queue: People serving themselves at a wedding buffet — first in line eats

first.
Circular queue: Buses at Nyabugogo terminal loop back to the start after finishing their route.
Deque: Boarding a bus from either the front or rear doors — people can enter or exit from both ends.

Q8: How can queues model this process?

Answer: At a Kigali restaurant, customer orders are enqueued when placed. When the meal is ready, the order is dequeued, and the customer is served. This ensures food is delivered in the correct order.

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

Answer: At CHUK hospital, emergencies must be treated before non-emergencies, even if they arrived later. This is a priority queue since patients with higher urgency jump ahead in line.

Q10: How would queues fairly match drivers and students?

Answer: In moto/e-bike taxi apps, drivers and students are placed in queues. The system matches the first available driver with the first waiting student, ensuring fairness and efficiency. Neither side can skip ahead of the other.