COLLEGE OF BUSINESS AND ECONOMICS

NAMES: IRADUKUNDA Marie Louise

Reg No:224009168

DEPARTMENT: BIT

YEAR: LEVEL TWO

MODULE TITTLE:DATA STRUCTURE AND ALGORITHM

EXSERCISES 2: STACK AND QUEUE

Part I – STACK

Q1: The MoMo app shows the LIFO idea because when you are filling steps like amount, account number, or PIN, the last step you entered is the first to be removed if you press back. This is exactly how a stack behaves: the top item is the first to come out, while older steps remain inside until their turn comes.

Q2: On UR Canvas, when you press back, the most recent page or module you visited disappears first. This is like the Pop operation where the top element is removed from the stack, leaving the earlier ones untouched. It shows how actions are undone in reverse order of how they were done.

Q3: In mobile banking, every new action such as sending money is pushed onto the stack. If you make a mistake, the undo function can simply pop the last action from the stack, removing the wrong step and keeping all the earlier correct transactions safe.

Q4: Stacks can ensure that forms like those on Irembo are balanced. Each time an opening bracket or field is added, it is pushed onto the stack, and it must be closed properly before being popped. If one remains unmatched, the stack will not be empty, showing an error in the form.

Q5: The sequence was Push (“CBE notes”), Push(“Math revision”), Push(“Debate”), Pop(), Push(“Group assignment”). After popping “Debate,” the top becomes “Group assignment.” This means the student will now deal with “Group assignment” next before touching older tasks.

• Push: adds an element to the stack.

• Pop: removes an element from the stack.

• Peek: display at top element of the stack

Q6: If a student undoes three actions, the last three items added are popped from the stack. This leaves only the earlier answers, meaning the stack now shows the work that was done before the last three steps were performed. It resets the state to an older version.

Q7: In booking tickets on RwandAir, each form step (personal details, flight choice, payment) is pushed onto the stack. When the passenger clicks back, the stack pops those steps in reverse order, allowing them to retrace the exact path step by step without skipping.

Q8: To reverse the proverb “Umwana ni umutware,” each word is pushed onto the stack in order. When they are popped out, they come out in the opposite order, forming “umutware ni Umwana.” This shows how powerful stacks are in reversing sequences.

Q9: A stack fits DFS (Depth First Search) because the algorithm goes deep into one branch before moving to another. With stacks, the last visited node is explored first, making it easier to backtrack. A queue, on the other hand, works level by level (BFS), not deep.

Q10: In the BK app, a stack could allow users to move back through transactions one step at a time. Each viewed transaction is pushed and popping them lets the user move back to earlier transactions easily, helping them track history in a controlled way.

Part II – QUEUE

Q1: In a Kigali restaurant, the first customer to arrive is the first one to be served. This is FIFO behavior because the person who joined earliest gets service before those who arrived later. A queue works exactly like this in computer systems.

Q2: A YouTube playlist works like a queue because the first video in the list is played and removed, then the next one comes automatically. This is just like the Dequeue operation, where the front item is taken away and the rest move forward in order.

Q3: At RRA offices, taxpayers line up and are served one after another. Each new person joins the back, and the front person is served first. This real-life situation behaves like a queue where Enqueue adds to the rear and Dequeue removes from the front.

Q4: Queues improve customer service because everyone is served in the order they arrive. This prevents unfair skipping and makes the process organized. Service centers like MTN or Airtel use queues to reduce confusion and to make sure no customer is ignored.

Q5: The sequence is Enqueue("Alice"), Enqueue("Eric"), Enqueue("Chantal"), Dequeue(), Enqueue("Jean"). When Alice leaves, the queue becomes [Eric, Chantal, Jean]. Therefore, Eric is now at the front, waiting to be served next.

• **Enqueue**: the enqueue operation inserts an item at the rear of the queue

• **Dequeue**: the dequeue operation deletes the item at the front of the queue

• **Display**: show elements in the array.

Q6: In RSSB pensions, the queue ensures fairness because files are treated in the exact order they were submitted. No one can jump ahead, so everyone's request is processed fairly, showing the importance of FIFO in real systems.

Q7: A linear queue is like people standing in a buffet line; service goes straight from front to rear. A circular queue is like buses looping at Nyabugogo—after reaching the last stop, they return to the beginning. A deque is like boarding a bus where you can enter from front or rear doors.

Q8: In a restaurant, when customers place food orders, each order is enqueued. When the food is ready, the order is dequeued and given to the correct customer. This ensures that meals are served in the same order as requests.

Q9: A priority queue is used at CHUK because emergencies must be treated immediately, even if other patients arrived earlier. Unlike a normal queue, service here is not based on arrival time but on urgency or importance.

Q10: In a moto app, students' requests are enqueued and drivers are matched in order. Queues make sure the first student waiting gets the first available driver, avoiding unfair skipping and keeping the system fair for both sides.