NAMES: DUFITIMANA OLIVIER

STUDENT NUMBER: 224005582

BIT YEAR TWO

DATA STRUCTURE AND ALGORTHMS

PROJECT 14

1.CHALLENGE AND REFLECTION QUESTION ON STACK

Q1. Challenge: Show how stack can reverse order of list [“Kigali”, “Musanze”, “Huye”].

A1. Here are the step-by-step instructions to reverse the list ["Kigali", "Musanze", "Huye"] using a stack:

Step1. Initialize an empty stack:

stack = []

Step2. Push "Kigali" onto the stack:

Stack.append ["Kigali"].

Step3. Push "Musanze" onto the stack:

Stack.append ["Musanze"].

Step4. Push "Huye" onto the stack:

Stack.append ["Huye"].

Step5. Pop elements from the stack

- Pop "Huye" (first pop): Resulting list is ["Huye"].

- Pop "Musanze" (second pop): Resulting list is ["Huye", "Musanze"].
- Pop "Kigali" (third pop): Resulting list is ["Huye", "Musanze", "Kigali"].

Step6. Final result:

The reversed list is ["Huye", "Musanze", "Kigali"].

This process uses the Last In, First Out (LIFO) property of a stack to reverse the order.

Q1. Reflection: Why does stack fit undo but not serving customers?

A stack fits the undo operation because it follows the Last In, First Out (LIFO) principle, where the most recent action (top of the stack) is undone first, mirroring how undo typically works. However, it doesn't suit customers well because serving customers often requires a First In, First Out (FIFO) or priority-based approach (e.g., a queue), where the order of service depends on arrival or priority, not just the last customer.

2.CHALLENGE AND REFLECTION QWUESTIONS ON QUEUE

Q1. Challenge: Create a queue model for UR library book requests. What issue arises is stack is used instead?

A1. **Step1**. Initialize an empty queue

Queue = []

Step2. Push first element into queue

Queue.append[BOOK REQUEST A]

Step3. Push second element into queue

Queue.append[BOOK REQUEST B]

Step4. Push third element into queue

Queue.append[BOOK REQUEST C]

Step5. Push fourth element into queue

Queue.append[BOOK REQUEST D]

Because in queue we FIFO first in first out then we use stack the system changes into LAST IN FIRST OUT.

When we use stack instead issues that arise will look like this:

- The student who requested book at first will be served last.

Queue. Pop ("BOOK REQUEST D")

Queue. Pop ("BOOK REQUEST C")

Queue. Pop ("BOOK REQUEST B")

Queue. Pop ("BOOK REQUEST A")

The following issues will arise: User Frustration, Inefficiency, Potential for Abuse, Loss of Order

Q2. Reflection: why is queue necessary in-service environment with multiple clients?

n a service environment with multiple clients, such as a library handling book requests, a queue is essential for managing requests efficiently, fairly, and predictably. Below, I reflect on the key reasons why a queue is necessary, drawing from the context of the University of Rochester (UR) library book request system and general service principles.

1. Ensures Fairness (First-In-First-Out, FIFO)

- Why it matters: In a service environment, clients expect their requests to be processed in the order they are submitted. A queue enforces FIFO, ensuring that the first client to make a request is the first to be served.
- Library example: If User A requests a book at 10:00 AM and User B at 10:05 AM, a queue processes User A's request first, aligning with fair expectations. Without a queue, later requests (e.g., in a stack) could be prioritized, causing frustration for earlier clients.
- Reflection: Fairness builds trust in the system. Clients are more likely to engage with a service they perceive as equitable, especially in a shared resource environment like a library where demand for books can exceed supply.