NAME:NAVIN GIRI

ROLL NO:ME24B1075

QUE N0:7

Hydroelectric Dam Flow Manager – Simulation Report

Objective

This project simulates the operation of a hydroelectric dam using core data structures in C – **Queue, Stack, Array, Singly Linked List, Doubly Linked List, and Circular Linked List**. The goal is to manage flow requests, emergency situations, power output logging, and component maintenance efficiently.

(a) Flow Request and Emergency Adjustments

Data Structures Used:

- **Queue** (FIFO) for normal flow requests.

- **Stack** (LIFO) for emergency adjustments.

Why These Structures?

- **Queue** is ideal for flow requests because it ensures the **first request gets handled first**, maintaining a natural order from sensors.
- **Stack** is perfect for emergencies because the **latest issue (e.g., valve pressure)** should be handled **first** for rapid response.

Code Summary:

- We enqueue six requests: "Turbine", "Gate", "Spillway", "Reservoir", "Pump", "Valve".
- As each is dequeued, it’s pushed onto a **stack**.
- Then we pop all from the stack to see the **reverse order of emergency handling**.

(b) Power Output Logging

Data Structure Used:

- **Array (Fixed-size, circular)** of 5 slots to store latest power outputs.

Why This Structure?

- An array is **fast and direct** to access.
- The circular buffer ensures **space efficiency**: when full, the **oldest output is transmitted** to avoid overflow.

Code Summary:

- We insert 7 outputs ("Pow1" to "Pow7") into a 5-slot array.

- After 5 insertions, new outputs start replacing the oldest, simulating a **rolling log**.
- Transmitted outputs are printed before replacement.

(c) Worn Component Maintenance

Data Structures Used:

- **Singly Linked List** for tracking **worn components**.
- **Doubly Linked List** for **repaired components**.

Why These Structures?

- Singly Linked List is **lightweight** and ideal for simple insertions (e.g., adding worn parts).

- Doubly Linked List allows **bidirectional traversal**, useful when reviewing repairs **forwards and backwards**.

Code Summary:

- "Turbine" and "Pump" are added to the worn list.
- "Turbine" is then **removed from worn** and added to the **repaired list**.
- The doubly linked list is then **traversed both ways** for review.

(d) High-Priority Tuning

Data Structure Used:

- **Circular Linked List** for urgent components like "Gate" and "Spillway".

Why This Structure?

- Circular linked lists are excellent when **cyclical or repeated monitoring** is needed.
- It ensures we can **loop over components endlessly** – perfect for continuously tuning high-priority parts.

Code Summary:

- We insert "Gate" and "Spillway" into a circular list.
- Then, we **traverse the list twice**, simulating repeated urgent checks or tuning cycles.

Conclusion

This simulation illustrates how **choosing the right data structure** simplifies and optimizes system management in real-world applications like hydroelectric dams:

Component	Structure Used	Benefit
Flow Requests	Queue	Maintains order from sensors (FIFO)
Emergency Fixes	Stack	Rapid last-in-first-out response (LIFO)
Power Logs	Circular Array	Efficient fixed storage, avoids overflow
Worn Components	Singly Linked List	Simple, lightweight tracking
Repaired Components	Doubly Linked List	Review forwards and backwards
Priority Tuning	Circular Linked List	Infinite cycling for urgent tasks

The program is modular and cleanly split into logical parts, making it ideal for extension (e.g., sensor data integration or graphical output) in future versions.