Earthquake Rescue Robot Coordinator using Data Structures

Name: [RATHOD JAISH]

Roll Number: [ME24B1041]

🔧 *Objective*:

To simulate a *Rescue Robot Coordination System* during an earthquake disaster using various data structures like Queue, Stack, Arrays, Singly & Doubly Linked Lists, and Circular Linked Lists. The aim is to manage rescue task assignment, mission logging, robot damage tracking, and urgent redeployment.

🧩 *System Components and Implementation*

✅ *a) Mission and Urgency Handling*

- *Queue* stores six predefined rescue tasks:

  {"Scanner", "Digger", "Lift", "Light", "Drone", "Cutter"}

- These tasks are dequeued and pushed into a *Stack* to simulate *LIFO* urgent task execution.

- *Output:*

  Tasks are handled in reverse order — showing urgency handling in *Last-In-First-Out* order.

📌 *Why LIFO?*

In real-time rescue, the most recently added task (like "Cutter") might be the *most urgent*, such as clearing life-threatening debris. LIFO ensures immediate attention to the most critical situations.

✅ *b) Rescue Log Unit*

- A *fixed-size array* with 6 slots is used to log completed missions.
- When 8 missions (Mis1 to Mis8) are logged, the oldest entries are removed as needed.
- *Overflow Handling:*

  When the array is full, the first element (Mis1, Mis2) is deleted to make space for new ones.

📌 *Why Log Missions?*

Logging is crucial for *tracking progress, **reporting survivor rescues, and **maintaining transparency* in rescue operations. Deleting the oldest keeps the log current.

✅ *c) Damaged & Repaired Robot Tracking*

- *Singly Linked List* tracks *damaged robots* like "Digger" and "Drone".
- "Digger" is repaired and moved to a *Doubly Linked List* to support *forward and backward inspection*.
- *Traversal*:

  - Forward and backward traversal of repaired robots is shown in output.

📌 *Damage and Fix Example:*

"Digger"'s arm was *crushed by falling debris* and later rebuilt with *reinforced hydraulics*, making it ready for reuse.

✅ *d) Priority Redeployment*

- *Circular Linked List* stores high-priority robots like "Scanner" and "Lift" for *urgent redeployment*.

- *Circular traversal* is repeated twice to simulate 24/7 patrol or re-checks.

📌 *Upgrade for Priority Bots:*

- "Lift" is equipped with a *thermal camera* to locate survivors.

- "Scanner" now includes *air quality sensors* to detect breath signatures or toxic gas.

📄 *Sample Output Overview*

Let's add the 6 tasks to the queue:

Urgent task being handled: Cutter

Urgent task being handled: Drone

Urgent task being handled: Light

Urgent task being handled: Lift

Urgent task being handled: Digger

Urgent task being handled: Scanner

Logging 8 missions:

Log's full. Deleting the oldest mission: Mis1

Log's full. Deleting the oldest mission: Mis2

Repaired Robots (Forward): Digger

Repaired Robots (Backward): Digger

High Priority Robot Redeployment:

Scanner -> Lift -> (loop 1)

Scanner -> Lift -> (loop 2)

📌 *Conclusion*

This program successfully simulates a complex *rescue robot coordination system* using basic data structures in C. It demonstrates how Queue, Stack, Arrays, and Linked Lists can solve real-life problems in disaster management.

✅ *Key Learnings*

- Efficient task handling using *Queue & Stack*

- Managing memory-limited logs using *Array overflow logic*

- Robot status tracking using *Singly & Doubly Linked Lists*

- Circular robot redeployment for *continuous mission loops*

Would you like this report as a downloadable file (PDF/Word)? I can help with that too!