EDSA ASSIGNMENT REPORT

Venkata Shruthi Pullela
ME24B1064

What problem is being solved ?

How to coordinate multiple autonomous rescue robots effectively during an earthquake disaster scenario. After an earthquake, time is critical. Survivors may be trapped, paths are blocked, and resources are limited. Human rescuers can't reach every area quickly or safely — so robots are deployed. However, managing all these different robot types (like Diggers, Drones, Scanners) and tasks efficiently in real-time is a complex coordination challenge.

Key objectives

Task Prioritization
Normal tasks are queued (FIFO).
Urgent tasks use a LIFO stack for immediate action, prioritizing the most recent emergencies.

Mission Logging
Completed missions are logged in a 6-slot array.
When full, the oldest logs are reported and replaced to keep updates current.

Robot Repairs
Damaged robots go into a singly linked list.
Repaired robots move to a doubly linked list for inspection, then to a circular list for quick redeployment.

Urgent Redeployment
Critical robots (e.g., “Scanner”, “Lift”) loop through a circular list, enabling continuous high-priority missions.

Why specific data structure was choose and how they help solve the problem

1. Queue (FIFO) – For Normal Mission Requests(optimal task ordering.)

Why this structure?

A queue stores tasks in the order they arrive, ensuring fairness.

How it helps:

- Ensures that rescue requests are handled in the same sequence survivors send them.
- Prevents task skipping or bias.
- Matches real-world line-up logic (e.g., help is dispatched in order).

2. Stack (LIFO) – For Urgent Task Handling(optimal task ordering.)

Why this structure?

A stack processes the most recent urgent task first, which is ideal in emergencies.

⚙️ How it helps:

- Ensures critical last-minute threats (e.g., sudden collapse) are addressed immediately.
- Mirrors real rescue scenarios where newer threats override older ones in urgency.

3. Fixed-Size Array – For Rescue Mission Logs(bounded log management.)

Why this structure?

An array with 6 slots is simple, fast, and reflects limited memory like in black-box systems.

How it helps:

- Quick access and overwrite capabilities.
- Automatically discards old logs to make room for new ones.
- Efficient for a rolling log of the most recent activity.

4. Singly Linked List – For Damaged Robots(dynamic tracking and flexible updates.)

Why this structure?

A singly linked list is ideal for adding/removing items quickly from the head.

How it helps:

- Easily tracks incoming damaged robots.
- Minimal memory and fast insertion (O(1)) at the front.
- Suitable for temporary, one-way tracking (damage → repair).

5. Doubly Linked List – For Repaired Robots(dynamic tracking and flexible updates.)
Why this structure?

Allows both forward and backward traversal through the inspection process.

How it helps:

- Robots can be checked in order of repair or reverse for follow-up.
- Enables flexible navigation during inspection or verification stages.

6. Circular Linked List – For Priority Redeployment(dynamic tracking and flexible updates.)
Why this structure?

A circular list enables endless cycling, perfect for recurring urgent use.

How it helps:

- Repaired high-priority robots (e.g., “Scanner”) can re-enter missions continuously.
- Prevents resetting or re-adding each time.
- Efficient for simulating rotation of mission-critical units.

Variables and functions used and Logic

Rescue Log (rescueLog): A fixed-size array (6 slots) to store mission logs.

Damaged Robot Tracker: A singly linked list (Node) to store the names of damaged robots.

Repaired Robots: A doubly linked list (DNode) to track repaired robots, allowing traversal both forwards and backwards.

Priority Redeployment (circularHead): A circular linked list (CNode) to manage robots assigned to priority missions (these robots cycle in a circular pattern).

Stack (stack): A fixed-size stack (6 slots) for handling urgent tasks, following a LIFO (Last In, First Out) order.

Queue (queue): A fixed-size queue (6 slots) to store mission requests, following a FIFO (First In, First Out) order.

addDamagedRobot(name): Adds a robot to the `damagedHead` linked list when it's damaged.

removeDamagedRobot(name): Removes a robot from the damaged list when it's no longer damaged.

addRepairedRobot(name): Adds a robot to the `repairedHead` doubly linked list when it's repaired.

traverseForward(): Prints the list of repaired robots from the head to the tail of the doubly linked list.

traverseBackward(): Prints the list of repaired robots from the tail to the head of the doubly linked list.

addToCircularList(name): Adds a robot to the circular linked list (used for priority deployment).

traverseCircularList(rounds, n): Traverses the circular list `rounds * n` times, printing the names of robots in the circular pattern.

push(task): Pushes a task onto the stack (LIFO).

pop(): Pops a task from the stack (LIFO).

enqueue(task): Adds a task to the queue (FIFO).

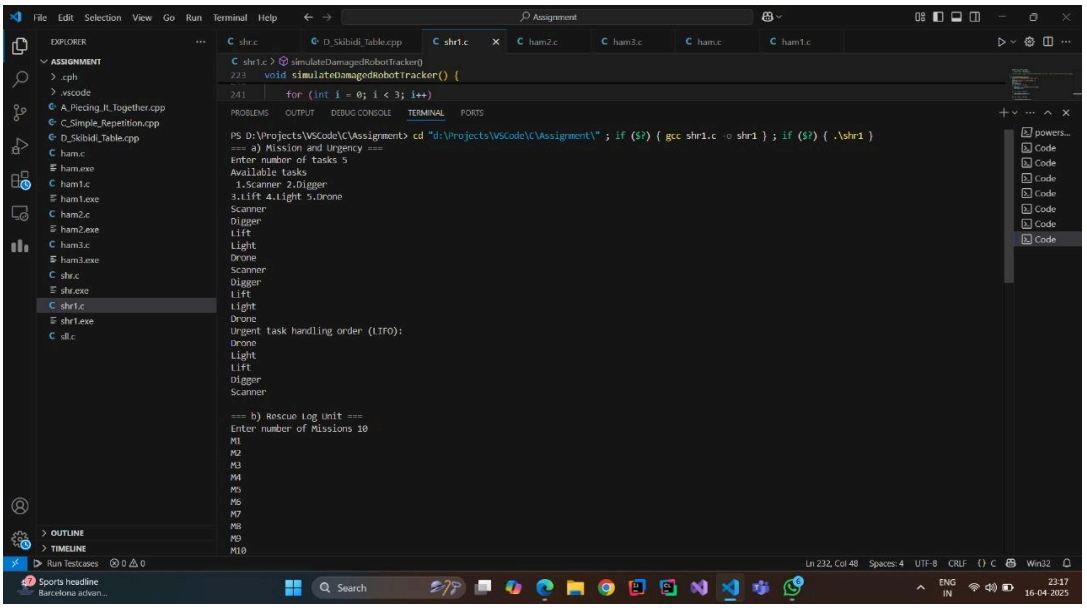
dequeue(): Removes and returns a task from the queue (FIFO).

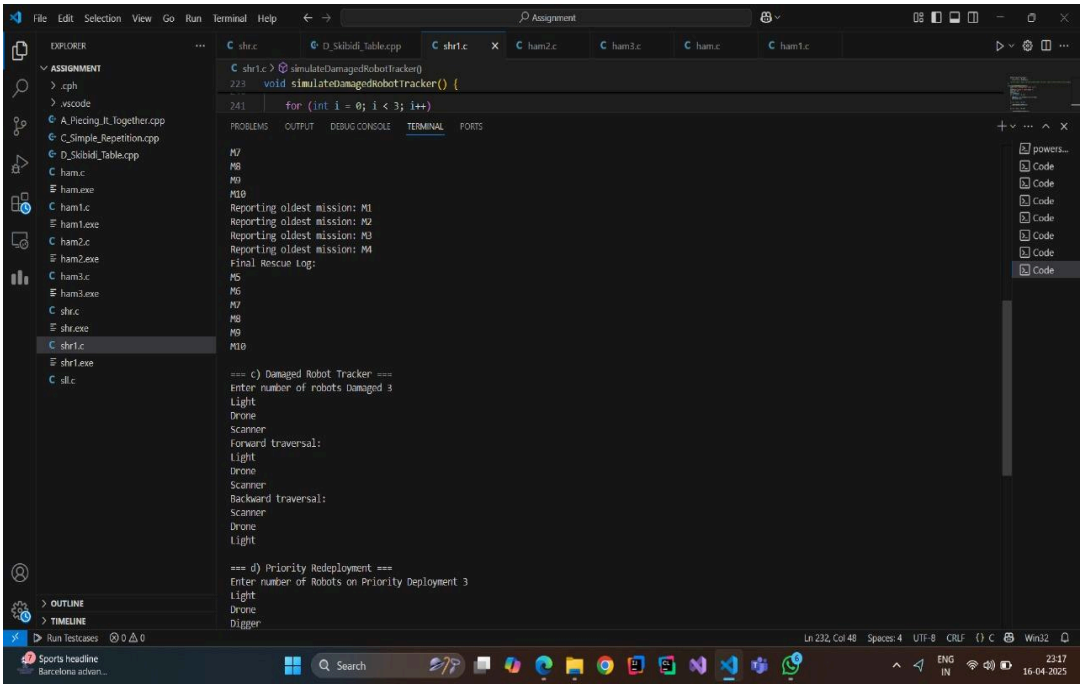
logMission(mission): Logs a mission by adding it to the `rescueLog` array. If the log is full, it shifts the array to remove the oldest mission.

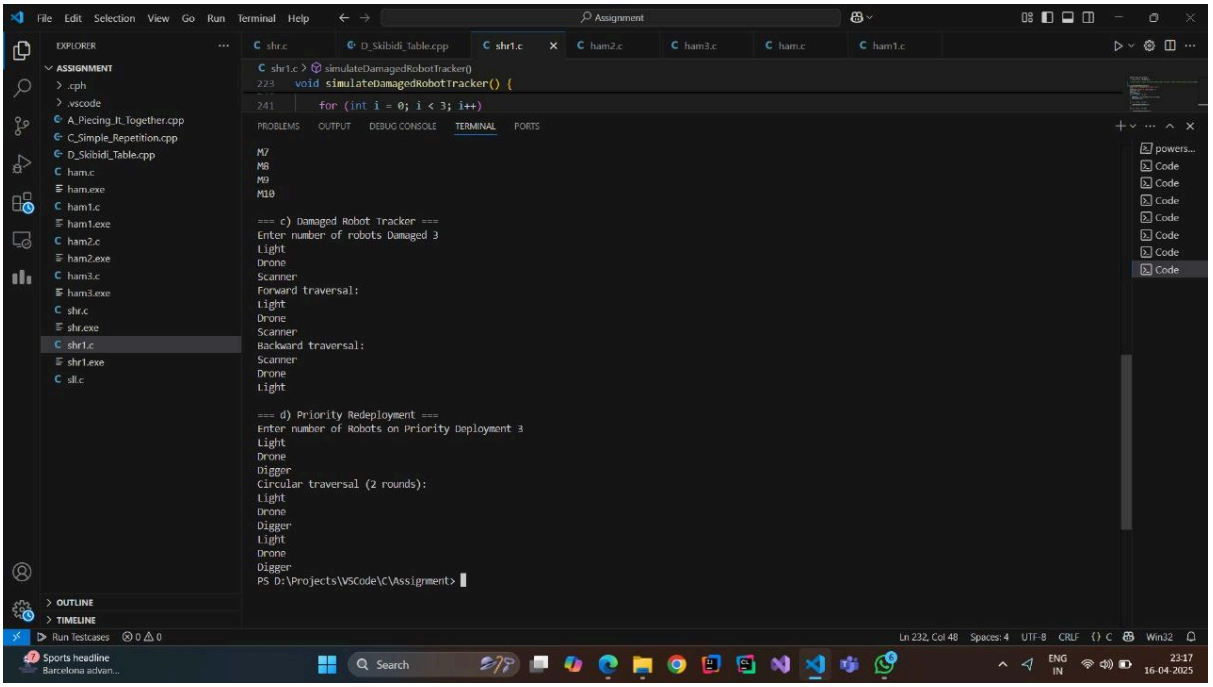
Stimulation function's

1. simulateMissionAndUrgency():
   - Simulates urgent mission handling by first adding tasks to the queue and then pushing them to the stack.
   - Tasks are processed from the stack in LIFO order (most recent first), which is useful for urgent tasks.
2. simulateRescueLogUnit():
   - Simulates logging rescue missions. Missions are entered by the user and logged to the `rescueLog`.
   - If the log is full, the oldest mission is removed to make space for the new one.
3. simulateDamagedRobotTracker():
   - Simulates tracking damaged robots by accepting the list of damaged robots.
   - The damaged robots are added to the linked list. The first 3 robots are moved from the damaged list to the repaired list.
   - It then traverses the repaired robots list both forwards and backwards.
4. simulatePriorityRedeployment():
   - Simulates priority redeployment by adding robots to the circular linked list.
   - These robots are cycled through in the circular list for a number of rounds (`rounds * n`).

Code Output:


PS D:\Projects\VSCode\C\Assignment> cd "d:\Projects\VSCode\C\Assignment\" ; if ($?) { gcc shr1.c -o shr1 } ; if ($?) { .\shr1 }
=== a) Mission and Urgency ===
Enter number of tasks 5
Available tasks
1.Scanner 2.Digger
3.Lift 4.Light 5.Drone
Scanner
Digger
Lift
Light
Drone
Scanner
Digger
Lift
Light
Drone
Urgent task handling order (LIFO):
Drone
Light
Lift
Digger
Scanner
=== b) Rescue Log Unit ===
Enter number of Missions 10
M1
M2
M3
M4
M5
M6
M7
M8
M9
M10


M7
M8
M9
M10
Reporting oldest mission: M1
Reporting oldest mission: M2
Reporting oldest mission: M3
Reporting oldest mission: M4
Final Rescue Log:
M5
M6
M7
M8
M9
M10
=== c) Damaged Robot Tracker ===
Enter number of robots Damaged 3
Light
Drone
Scanner
Forward traversal:
Light
Drone
Scanner
Backward traversal:
Scanner
Drone
Light
=== d) Priority Redeployment ===
Enter number of Robots on Priority Deployment 3
Light
Drone
Digger


File Edit Selection View Go Run Terminal Help
Assignment
EXPLORER
ASSIGNMENT
.cph
.vscode
A_Piecing_It_Together.cpp
C_Simple_Repetition.cpp
D_Skibidi_Table.cpp
ham.c
ham.exe
ham1.c
ham1.exe
ham2.c
ham2.exe
ham3.c
ham3.exe
shr.c
shr.exe
shr1.c
shr1.exe
sll.c
OUTLINE
TIMELINE
shr.c
D_Skibidi_Table.cpp
shr1.c
ham2.c
ham3.c
ham.c
ham1.c
shr1.c > simulateDamagedRobotTracker()
223 void simulateDamagedRobotTracker() {
241 for (int i = 0; i < 3; i++)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
M7
M8
M9
M10
=== c) Damaged Robot Tracker ===
Enter number of robots Damaged 3
Light
Drone
Scanner
Forward traversal:
Light
Drone
Scanner
Backward traversal:
Scanner
Drone
Light
=== d) Priority Redeployment ===
Enter number of Robots on Priority Deployment 3
Light
Drone
Digger
Circular traversal (2 rounds):
Light
Drone
Digger
Light
Drone
Digger
PS D:\Projects\VSCode\C\Assignment>
powers...
Code
Code
Code
Code
Code
Code
Code
Run Testcases
Ln 232, Col 48
Spaces: 4
UTF-8
CRLF
C
Win32
Sports headline
Barcelona advan...
Search
ENG
IN
23:17
16-04-2025