Aisle Movement Simulation — Comparative Report

1. Problem Description

The simulation models movement of individuals represented by 1s along an aisle represented by 0s.

The algorithm must detect when a new person begins moving, detect continuation of a movement

chain, correctly identify boundary finishes, record unique IDs for all movers, and behave symmetrically for left and right movements.

2. Explanation of the Code Versions

Code with minimal ChatGPT modifications: A faithful reflection of the original logic. Works on simple cases but fails to classify movement chains correctly and misreports finishing identities.

Minor code enhancements (on "Code with minimal ChatGPT modifications"): Fix IDprinting and boundary-recording errors.

Code with most ChatGPT modifications: Implements complete chain detection, boundary handling,

ID tracking and full correctness.

3. Test Case Overview

A) [1,1,1,1,1] — No movement possible. All versions behave identically.

B) [0,0,1,0,0,0,1,0] — Complex multi-step behaviour. Only the full version is correct.

C) [1,1,1,1,1] — Repeated to confirm no state leakage.

4. High-Level Definition Summary

Purpose: simulate orderly movement while maintaining identity tracking and correct movement classification.

Key logical requirements include distinguishing NEW vs CONTINUING movements, tracking boundary finishes, registering unique IDs and maintaining left/right symmetry.

Aisle Movement Simulation — Comparative Report (Technical Analysis)

PURPOSE OF THIS DOCUMENT

This section provides a complete technical comparison between the three versions of the aisle movement engine:

• Code with minimal ChatGPT modifications

• Minor code enhancements on "Code with minimal ChatGPT modifications"

• Code with most ChatGPT modifications

It explains why the versions behave differently, which logical structures were missing, and how the patches synchronize behaviour.

HIGH-LEVEL DIFFERENCE SUMMARY

Code with minimal ChatGPT modifications:

• Behaves correctly for simple cases.

• Fails to detect continuation of movement chains.

• Incorrect ID storage before patch.

• Sometimes under-counts actual movers.

Code with most ChatGPT modifications:

• Fully correct continuation logic.

• Accurate new-person detection.
• Correct boundary handling.

• Correct ID registration and mover counting.

• Represents the intended design.

• <u>Minor code enhancements on “Code with minimal ChatGPT modifications”</u>

• Fixes ID-tracking issue.

• Improves finish detection.

• Still lacks full chain-handling.

<u>ANALYSIS OF TEST CASES</u>

Test Case A — [1,1,1,1,1]

• All versions PASS. No movement possible.

Meaning:
• Fully packed aisle.
• No zeroes available.
• No person can move.
• Only termination and scan-completion logic is tested. Why It Is Special:
• Verifies that no phantom movement occurs.
• Ensures code exits correctly after one scan.
• Tests correct handling of average calculations with 0 movers.
• Ensures no fake "new person" initiations. Expected Behaviour:
• Number of people moved: 0
• Total moves: 0
• Average: NaN
• Verdict: “no difference moving left or right”

<u>Code with most ChatGPT modifications:</u> Result:
PASS

<u>Minor code enhancements on “Code with minimal ChatGPT modifications”</u>
PASS (after patch) Reason:
No movement logic is triggered; both versions behave identically.

Test Case B — [0,0,1,0,0,0,1,0]

Most important scenario. Activates:

Only the code with most ChatGPT modifications passes fully.

Meaning:
• Two people, far apart.
• Each must move multiple times to reach the boundary.
• Tests ALL major behaviours:
- new person starts
- continuing movement
- previous-person finishing
- boundary finishing
- scan tracking
- ID registration
- correct recognition of single vs chained movement Why It Is Special:
• This is the most complete functional test.
• It activates:
- movement detection
- multi-step chaining
- finishing at boundary
- simultaneous behaviour from both ends
• Catches flaws in logic that simple cases do not expose. Expected Behaviour:
• Exactly 2 people should be recorded as movers.
• Movement chains must NOT be misinterpreted as new people.
• ID registration must list correct finishing positions.

<u>Code with most ChatGPT modifications:</u>
Result: PASS
- Tracks exactly 2 people moving.
- Correct chain interpretation.
- Correct finish/ID logs.
- Fully symmetrical behaviour.

<u>Code with minimal ChatGPT modifications:</u>
FAIL (without patch)
- Incorrect ID storage (0 instead of finishPoint).
- Incorrect detection of chain continuation.
- Under-counts number of people moved.
- Prints misleading logs.

Minor code enhancements on “Code with minimal ChatGPT modifications:
IMPROVED BUT STILL NOT PERFECT
- ID storage fixed.
- But chain logic still incomplete.
- May misclassify chain continuations as new movers.

Test Case C — [1,1,1,1,1]

Ensures no state leakage.

This is a repeat of Test Case A to confirm:
• Symmetry of behaviour before and after Test Case B.
• No state leaks across multiple runs.
• No accidental carry-over of flags.

Code with most ChatGPT modifications:
Result: PASS

Code with minimal ChatGPT modifications:
Result: PASS
Both behave identically when no movement is possible.

WHY TEST CASE B REVEALS FAILURES

The minimal version oversimplifies state management, leading to:

• Code with minimal ChatGPT modifications simplifies the rule:
“If a person moves and hasPersonPrevMove is false, treat it as a new start.” This leads to:
• Incorrect new-person detection.
• Incorrect boundary logic.
• Incorrect previous-person finishing.
• Incorrect counting of total movers.

Code with most ChatGPT modifications fixes this by introducing additional guarding logic such as:
- hasPreviousPerson
- hasPersonFinishedMoving
- currentNumberScansRow - scan-boundary verification These additions prevent:
• Phantom new starts,
• Multiple "beginnings" for the same continuous movement,

• And incorrect ID registration
•Incorrect continuation detection

The fully modified version includes deeper checks such as:

hasPreviousPerson
hasPersonFinishedMoving
currentNumberScansRow
scan boundary verification

5. Details of Minor Code Enhancements

PATCH 1 — Add missing “REGISTERING UNIQUE ID…” output
This patch reinstates a critical print statement that confirms a person’s identity at the moment they finish moving. Without this print, transitions appeared abrupt and misleading even though the logic stored IDs correctly. The missing line caused misunderstanding in alternating patterns, as the system visually appeared to start a new person before finishing the previous one.

```
System.out.println("PERSON FINISHED MOVING
AT POSITION: " + i); hasPersonFinishedMoving
= true; //--------------CHATGPT-----------
System.out.println("REGISTERING UNIQUE ID FOR PERSON
MOVED ONTO: " + i);
//--------------END
CHATGPT----------------------------
```

PATCH 2 — Fix incorrect ID storage on boundary finishes Originally, finishing a movement at the boundary always stored ID = 0, even when the correct finish index was the far-right boundary. This corrupted identity tracking and caused phantom “same person continuing” errors.

```
//--------------CHATGPT----------recordPersonMoved[k] =
finishPoint; //--------------END
               CHATGPT----------------------------
```

Expanded Patch Explanations

Why Patch 1 was critical

• Missing confirmation of identity registration.
• Confusion in logs such as “new person starts before previous finishes.”
• Ambiguous transitions in alternating patterns.

Why Patch 2 was critical

• Corrupted identity history.
• False “same person continuing” detections.
• Broken sequencing in alternating test cases.

Summary
These two patches were the ONLY corrections needed to bring the minimal version into alignment with the fully modified version. Everything else was structurally identical.

6. Final Conclusion

The corrected version remains the authoritative engine. The Minor code enhancements on “Code with minimal ChatGPT modifications” version is usable for reference, but lacks full chain-handling. Test Case B should always be used as the regression test for correctness.

• Code with most ChatGPT modifications is the definitive, correct implementation.
• Code with minimal ChatGPT modifications should only be used in patched form and only as a simplified reference.
• Test Case B is the gold-standard validation case: any future code revision must pass it to be considered correct.
• Test Cases A and C verify proper termination, stability, and absence of phantom behaviour

6. ☑ Code with most ChatGPT modifications **is STILL the correct engine**

Minor code enhancements on “Code with minimal ChatGPT modifications” fixes only the two bugs:

✓ Patch 1 — missing “REGISTERING UNIQUE ID…” print

✓ Patch 2 — incorrect ID assignment (0 instead of finishPoint)

These only repair **logging and identity corruption**, but **they do NOT fix the deeper structural limitations in** Code with minimal ChatGPT modifications, such as:

---

7. ◇ **Remaining limitations in** Code with minimal ChatGPT modifications **(even after patches)**

1️⃣ Minor code enhancements on “Code with minimal ChatGPT modifications” **does not have full chain-continuation detection**

Code with most ChatGPT modifications uses multiple flags:

- hasPreviousPerson
- hasPersonFinishedMoving
- currentNumberScansRow
- scan tracking depth

Minor code enhancements on “Code with minimal ChatGPT modifications” does **not track all internal states**, so it misinterprets:

- Multi-step movement chains
- Alternating movement patterns
- People finishing mid-scan
- People continuing across scans

2️⃣ Minor code enhancements on “Code with minimal ChatGPT modifications” **still cannot distinguish between:**

- New person start
- Same person continuing
- Boundary continuation
- Mid-aisle termination events

Code with most ChatGPT modifications does.

3️⃣ Minor code enhancements on “Code with minimal ChatGPT modifications” **lacks complete symmetric behaviour** Test case B exposes this directly.

4️⃣ Minor code enhancements on “Code with minimal ChatGPT modifications” **has no proper state-reset mechanism between scans**

Code with most ChatGPT modifications resets and transitions correctly.

5️⃣ Minor code enhancements on “Code with minimal ChatGPT modifications” **frequently overcounts or undercounts movers**

Your documentation shows this repeatedly.

6️⃣ Minor code enhancements on “Code with minimal ChatGPT modifications” **was never designed to handle “alternating movement” scenarios**

Only Code with most ChatGPT modifications behaves correctly on these.

---

8. ▨ **Conclusion (Single Sentence):**

✓ **Even with the correct patches applied,** Minor code enhancements on “Code with minimal ChatGPT modifications” **is NOT equivalent to** Code with most ChatGPT modifications**.**

✓ Code with most ChatGPT modifications **remains the *only* fully correct representation of your movement engine.**

---

9. 🧠 **Why your patched** Minor code enhancements on “Code with minimal ChatGPT modifications” **is still useful** It

is extremely valuable for:

- Educational comparison
- Visualising your earlier logic

- Understanding where tracking failed
- Demonstrating how the chain-detection rules evolved

- Cleanly illustrating the *difference* between partial and full correctness

But it is **not** the engine you should rely on for correctness.
