Student name: Sk Md Shariful Islam Arafat	**Student ID:** a1983627
Problem name: WhiteShoes	**Solution number - 1**

Q1.1. What are the errors/issues you identified and what is your recommendations/suggestions to improve the solutions to pass all the test cases? Please describe your approach to write the solution for the given problem. In case you used ChatGPT, what are errors, recommendations/suggestions ChatGPT provided to improve the solution to pass all the test cases? what is your takeaway from these identified errors and recommendations?

Errors/Issues Identified

- Relies only on the maximum count, missing overall consistency.
- Uses ad-hoc case checks (all zero, all white, etc.) that don't cover all scenarios.
- Fails cases like `[1,1,1,2]` because it ignores the key invariant.

Recommendations/Suggestions

- Use the mathematical invariant:
  - Compute `S = sum(count)`.
  - If `S % (n − 1) != 0`, return `−1`.
  - Otherwise, let `W = S / (n − 1)`. Validate `0 ≤ W ≤ n`.
- Verify each entry is either `W` (black shoe wearer sees all whites) or `W − 1` (white shoe wearer sees all others except themselves).
- Check exactly `W` entries equal `W − 1`. If not, return `−1`.
- This approach eliminates guesswork, handles all corner cases, and remains O(n).

My Approach

I started from the observation that each white shoe wearer is seen by everyone else, so total counts must equal `W∗(n−1)`. That gave me a formula to calculate `W`. From there, I validated individuals' counts against the expected pattern. This leads to a clean, rigorous solution.

Takeaway from Errors/Recommendations

When using ChatGPT, it pointed out:

- The reliance on `count[n−1]` was the root problem.
- It recommended replacing the "max guess" logic with the invariant `S = W∗(n−1)` and the validation check for each entry.
- It also showed how to restructure the code for clarity and correctness.

Takeaway: Relying on mathematical invariants is more reliable than case-by-case reasoning. Automated tools can help spot missing general conditions, but I need to critically validate the suggestions and connect them back to the problem definition. This ensures I understand why the fix works, not just that it works.

Q1.2. What challenges have you faced in identifying a complete solution to the given problem? What type of help would you need to solve similar problems like this? What kind of feedback would you like to receive from your teaching team to improve your programming skills?

Challenges Faced

The main challenge was realizing that checking only the maximum value or handling special cases was not enough. It was difficult to see the global condition (`sum(count) = W*(n-1)`) that guarantees correctness for all inputs.

Help Needed

For similar problems, I would need guidance on how to identify mathematical patterns or invariants in counting/consistency problems. Example hints or walkthroughs of smaller test cases would also help.

Feedback Wanted from Teaching Team

I would like feedback that not only points out mistakes but also explains the reasoning behind the correct approach. Clear examples of how to connect problem constraints to code logic would improve my problem-solving and debugging skills.

Q1.3. What are your suggestions to improve the feedback provided by AI tools like ChatGPT to cater to your learning needs and to guide you in programming tasks which would help you to be a skilled programmer?

Suggestions to Improve AI Feedback (e.g., ChatGPT)

AI tools like ChatGPT should:

- Provide step-by-step reasoning instead of only showing the final code.
- Explain *why* a condition or formula works with small examples.
- Highlight common mistakes (like relying on max values) and show better patterns (like invariants).
- Suggest practice exercises with increasing difficulty to strengthen concepts.
- Give feedback in a structured way: what's wrong, why it fails, and how to fix it.

This kind of guidance would make the feedback more educational and help me grow as a programmer instead of just copying solutions.

Problem name: WhiteShoes	**Solution number - 2**

Q2.1. What are the errors/issues you identified and what is your recommendations/suggestions to improve the solutions to pass all the test cases? Please describe your approach to write the solution for the given problem. In case you used ChatGPT, what are errors, recommendations/suggestions ChatGPT provided to improve the solution to pass all the test cases? What is your takeaway from these identified errors and recommendations?

Errors/Issues Identified

- **Wrong invariant:** Uses `sum/2` logic; the correct relation is `S = W*(n-1)` where `S = sum(count)` and `W` is #white.
- **Bad bounds check:** Uses `count[i] > count.length`; the max anyone can see is `n-1`.
- **No per-entry validation:** Does not enforce each `count[i]` ∈ {`W-1`, `W`}.
- **No composition check:** Does not ensure exactly `W` entries equal `W-1` (the white wearers).
- **Irrelevant guard:** `cnt > count.length * 2` is not tied to the real invariant.

Recommendations/Suggestions

- Compute `S = sum(count)`; if `S % (n-1) != 0` → `-1`.
- Set `W = S/(n-1)`; validate `0 ≤ W ≤ n`.
- Check all `count[i]` ∈ {`W-1`, `W`} and that exactly `W` entries are `W-1`.
- Keep O(n) with one pass for sum and one pass for validation.

Approach to Write the Solution

Derive the global invariant (`S = W*(n-1)`) from "each white is seen by all other people," then validate individuals' counts and the exact number of white wearers.

ChatGPT Feedback & Takeaway

ChatGPT recommended replacing `sum/2` with `S = W*(n-1)`, fixing the bound check to `n-1`, and enforcing {`W-1`, `W`} with exactly `W` occurrences of `W-1`.

Takeaway: Favor provable invariants + local validation over heuristics; encode constraints directly to cover all edge cases.

Q2.2. What challenges have you faced in identifying a complete solution to the given problem? What type of help would you need to solve similar problems like this? What kind of feedback would you like to receive from your teaching team to improve your programming skills?

Challenges Faced

- Recognizing the correct invariant was hard: I initially modelled counts as "pairs" (`sum/2`) instead of the true relation `S = W*(n-1)`.
- Mapping local counts to roles (white sees `W-1`, black sees `W`) and proving that exactly `W` entries must be `W-1` was non-obvious.
- Building counterexamples to stress the logic (e.g., mixed 7/8 array) took time.

Help Needed

- Hints on deriving invariants from problem statements and translating them into code checks.
- Short guided exercises that practice consistency constraints and proof-by-contradiction with small cases.
- Examples showing how to move from a wrong model (pair counting) to the right model.

Feedback I'd Like from Teaching Team

- Targeted feedback that explains **why** a model is incorrect and how to construct the correct invariant.
- Clear checklists: required bounds, divisibility checks, and per-element validations.
- Suggestions for incremental tests to validate edge cases and prevent heuristic shortcuts.

Q2.3. What are your suggestions to improve the feedback provided by AI tools like ChatGPT to cater to your learning needs and to guide you in programming tasks which would help you to be a skilled programmer?

Suggestions to Improve AI Feedback

- AI should explain *why* an assumption (like `sum/2`) is wrong and show the correct invariant with examples.
- Feedback should include **step-by-step reasoning** before showing code, so I learn the thought process.
- Provide **common pitfalls** (e.g., confusing pair counting vs. global consistency) to avoid repeating mistakes.
- Offer **mini test cases** to check logic and encourage me to debug with those.
- Structure responses as: *identify issue → explain concept → show fix → suggest practice*.

This way, AI feedback not only corrects the solution but also strengthens my problem-solving skills and helps me build confidence for similar tasks.

Write your solution below:

```java
public class WhiteShoes {

    public static int howMany(int[] count) {
        int n = count.length;
        int sum = 0;

        // Check each value is within valid range
        for (int c : count) {
            if (c < 0 || c > n - 1) return -1;
            sum += c;
        }

        // Total must match W * (n - 1)
        if (sum % (n - 1) != 0) return -1;
        int W = sum / (n - 1);

        // W must be within bounds
        if (W < 0 || W > n) return -1;

        // Validate counts: only W or W-1 are allowed
        int whitesSeen = 0;
        for (int c : count) {
            if (c == W - 1) {
                whitesSeen++;
            } else if (c == W) {
                // valid
            } else {
                return -1;
            }
        }

        // Exactly W entries must equal W-1 (the white shoes wearers)
        return whitesSeen == W ? W : -1;
    }
}
```