Assignment

CA1: Application of Concurrency to Common Tasks

Name	**Diyan Kashif**
Roll Number	**2022526**

GitHub Repository: https://github.com/2022526a/ConcurrentSystem

Task 1: Standard Deviation Calculation

Pros of Concurrency:

- If you've got a ton of data, splitting the work speeds things up (like group projects but actually helpful).
- Uses all those fancy CPU cores your laptop bragged about when you bought it.
- Stops your program from freezing like an old PC when you're running heavy stats.

Cons of Concurrency:

- For tiny datasets, creating threads is like hiring a team to eat one chip—pointless overhead.
- Shared variables can turn into a mess if you're not careful (*"Wait, why is my result 12… or 47… or…?"*).
- Debugging? Imagine herding cats, but the cats are threads and they're all crashing.

Bottom Line:
Good for big data, overkill for small stuff. And *please* synchronize—or suffer.

Task 2: Matrix Multiplication

Pros of Concurrency:

- Each cell in the result matrix is like a mini math problem—solve them all at once!
- Huge matrices? Concurrency turns "forever" into "meh, a few seconds."
- CPUs love caching parallelized data (but good luck explaining why in the report).

Cons of Concurrency:

- You going to split tasks perfectly, or threads just sit around the bored group members.
- It uses the high memory usage.
- Too many threads = your CPU starts *switching* more than *working*.

Bottom Line:
Great for big matrices, but manage threads like a strict cafeteria monitor.

Task 3: Merge Sort (Descending)

Pros of Concurrency:

- Merge sort *wants* to be parallel—split data, sort chunks, merge. Easy!
- Big datasets get sorted way faster (like delegating chores to siblings).
- Threads can sort *and* merge simultaneously (if you don't mess it up).

Cons of Concurrency:

- Small datasets? Don't need this approach.
- Recursion + concurrency = stack overflow errors (like "Why is my program dead?").
- Merging needs locks, or threads fight like kids fight for the last cookie.

Bottom Line:
Perfect for big data, but control threads or chaos ensues.

CSV File Reading — Concurrency vs Single-Threaded

Pros of Concurrency:

- A lot of files parse faster (if your disk isn't a potato).
- UI stays responsive—no "Not Responding" errors.
- Producer-consumer flow feels fancy (like an assembly line for data).

Cons of Concurrency:

- Disks read sequentially, so concurrency sometimes helps as much as sunglasses at night.
- Parallel reads can cause file locks (*"Hey, I was using that!"*).
- Code gets messy—like spaghetti, but less tasty.

Bottom Line:
Only worth it for massive CSVs. Otherwise, single-thread and chill.

Final Thoughts

Concurrency is a **superpower** for heavy tasks (matrix math, big sorts), but it's like giving a toddler caffeine—**great energy, zero control**. Use it wisely:

- Big data? Yes.
- Small data? Skip the headache.
- *Always* test for race conditions, or your debug sessions will be existential crises.