Problem 1

Race Condition: The unexpected results are due to a race condition, where multiple threads try to read, modify, and write to a shared variable (someValue) concurrently without any coordination. This results in lost updates and inconsistent values.

Why It Gets Worse with More Iterations: As COUNT increases, the probability of concurrent access to someValue increases, leading to more frequent inconsistent modifications. This is why the discrepancy becomes larger when COUNT is increased to higher values.

Synchronization Solution: The race condition is fixed by using a mutex to ensure synchronized access to the shared variable. Mutexes provide a locking mechanism that ensures only one thread at a time can modify the shared variable, leading to consistent and predictable results.