CS5103: Assignment 10 // Date: 23-10-24

Question 1: "The Harvest Calculation"

The kingdom of Sumland is known for its vast and fertile fields that yield an abundant harvest each season. To manage their kingdom efficiently, the farmers of Sumland work in teams, each responsible for a part of the field. However, summing up the total harvest has always been a challenge.

The King of Sumland has tasked you, the Royal Programmer, with creating a system that divides the kingdom's harvest fields into equal parts. Each part will be assigned to a different team (represented as a thread). Your program must calculate the sum of each team's harvest independently, and then the Royal Accountant (the main thread) will combine these sums to find the total harvest output.

Objective:
Write a multithreaded program that divides an array (representing the harvest values) into equal parts and assigns each part to a separate thread. Each thread will compute the sum of its assigned part, and the main thread will combine these partial sums to determine the total sum.

Hint: Use global variables or pass structures to the threads. Ensure proper synchronization using pthread_join.

Test Case:
Enter number of fields: 16
Enter number of teams: 4
The harvest array contains 16 fields and is divided among 4 teams (threads).
Enter Array: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}

The total sum of the harvest should be:
Total sum: 136

Question 2: "The Synchronized Machinery"

The bustling city of Gearhold relies on a central machine that powers its factories. The machine’s efficiency depends on a shared counter that keeps track of the power cycles it completes. Multiple workers (represented as threads) are assigned to operate the machine, increasing the counter as they work. However, without proper synchronization, the machine could malfunction, leading to inaccurate counts and eventual breakdown.

As the Master Engineer, you must implement a solution that ensures the counter’s accuracy, even when multiple workers are incrementing and decrementing it simultaneously. To prevent data races and ensure thread safety, you’ll use a special lock mechanism (a mutex) to control access to the counter.

Objective:
Write a thread-safe multithreaded program where multiple threads increment a shared counter without causing any data races. Use pthread_mutex_t for synchronization to lock and unlock access to the shared counter safely.

Test Case:
Enter number of workers (threads): 5
Enter number of times each worker will increment the counter: 1000

Final counter value: 5000