Pthreads Excercises

1- Create a program so that each thread receives an integer argument representing its ID and prints "Thread X is running". Use pthread_create() to pass the ID inside a **struct ThreadData { int id; }**, ensuring each thread correctly handles its data.

2- Create a program with three threads where each thread computes the square of its ID. Store results inside a struct ThreadResult { int id; int square; }, and ensure the main thread waits for all threads to finish before printing their squared results.

3- **Parallel Sum:** Implement a program that calculates the sum of an integer array using two threads. Each thread should sum half of the array, using a **struct SumData { int* array; int start; int end; int result; }** to pass array information. The main thread collects both partial sums to compute the final result.

4- **Vector Addition:** Given two arrays of size N, create N threads, where each thread computes one element of the resulting sum array, C[i] = A[i] + B[i]. Define **struct VectorData { int* A; int* B; int* C; int index; }** to pass individual indices to the threads.

5- **Race Condition and Mutex:** Implement a shared counter that multiple threads increment concurrently. Define **struct Counter { int value; pthread_mutex_t lock; }**, and use a mutex inside the struct to synchronize access to the counter, preventing race conditions.

6- **Parallel Prime Number Finder:** Given a range [L, R], divide it among N threads so each thread finds prime numbers in its assigned subrange. Use **struct PrimeData { int start; int end; int* primes; int count; pthread_mutex_t lock; }** to manage thread-safe access to the shared prime number list.