Problem Set 8

Note: Try to make the best use of appropriate C++ features.

Single-thread vs Multithreaded Performance Comparison

This lab aims to demonstrate how multithreading can divide a computation task into multiple subtasks executed in parallel to significantly improve efficiency.
In this lab, you will process a large array of integers to **count the number of even numbers**, first with a **single-threaded** approach, then using a **multithreaded** approach, and finally compare the performance results.

Objectives:

- Learn the basics of multithreaded programming (std::thread, std::ref, join, etc.).
- Understand data partitioning and independent thread computation.
- Learn how to aggregate partial results from multiple threads.
- compare single-threaded and multithreaded performance and compute speedup.

Tasks:

Given a **600,000,000-element** (600 million) 1D array:

1. **Single-threaded Version**: size_t countEvenSingleThread(data)
   - Traverse the entire array using a single thread to count the number of even numbers.
2. **Multithreaded Version**: size_t countEvenMultiThread(data, numThreads)
   - Launch **n threads**.
   - Evenly divide the array into **n** segments, with each thread independently processing one segment.
   - Each thread stores its partial count in a local variable.
   - After all threads complete, aggregate the partial counts to obtain the total number of even numbers.
3. **Performance Comparison**:
   - Use the provided main function to test both implementations and evaluate the performance differences by different dataSize and numThreads.