COP 6616 Fall 2025

Programming Assignment 1

Submission Guidelines

- This is an **individual assignment**. Collaboration or group work is not permitted.
- You may use **any programming language** that supports multi-threading (preferably C, C++, Java, Rust).
- Submit your completed work via **Webcourses** only. Submissions sent by email will **not** be accepted.
- Make sure to include a **README file** with clear instructions on how to compile and run the program from the command line.
- **Due date: Wednesday, September 24th by 11:59 PM (ET)** Late submissions will **not** be accepted under any circumstances.

Part 1 — Parallel Prime Finder (Coding Component, 40 points)

Your non-technical manager has assigned you the task of finding all prime numbers between 1 and 10^8 . The company plans to use a parallel machine that supports **eight concurrent threads**, so your solution should spawn **8 threads** to perform the computation.

Although your manager doesn't have a strong technical background, she's a reasonable person. She expects the workload to be distributed so that **each thread completes its portion in approximately the same amount of time**.

Your company rents compute time by the minute, so execution speed directly affects cost. Choose your algorithm wisely.

Required Output

Your program should write the following to a file named `primes.txt`:

Code

```
<execution time>  <total number of primes found>  <sum of all primes found>
<top ten largest primes, listed in ascending order>
```

Output Guidelines:

- Do **not** include 0 or 1 in the prime count or sum, as they are neither prime nor composite.
- Measure execution time starting **before** thread creation and ending **after** all threads have completed.

Grading Breakdown

- **Program design and correctness** – 50%
- **Efficiency and performance** – 30%
- **Documentation and evaluation summary** – 20%

Additional Instructions

- Your solution must use a **parallel approach** and an **efficient prime-finding algorithm**.
- Cheating in any form will not be tolerated.
- Submit your work via **Webcourses**.

Part 2 — Analysis of Your Solution (Written Component, 60 points)

To demonstrate your understanding of the solution you submitted, please respond to the following questions in a separate written document (**PDF format**). This component is required and will be graded alongside your code submission.

Your responses should be clear, concise, and written in your own words. You may include diagrams, tables, or pseudocode if helpful.

1. Summary of Your Approach Briefly describe how your program finds prime numbers between 1 and 10^8 using parallel threads. Include:

- The overall structure of your solution
- How the workload is divided among threads
- Any libraries or frameworks used

2. Correctness and Efficiency Provide an informal explanation of why your program is correct and efficient. Address the following:

- How your algorithm ensures that only prime numbers are included
- How you avoid redundant or unnecessary computations
- How your design minimizes execution time and balances thread workloads

3. Experimental Evaluation Summarize the performance of your program based on actual execution. Your evaluation should include both your **8-threaded solution** and a **baseline single-threaded version** of your program. Address the following points:

- **Total execution time** for both versions
- **Number of primes found** and their **sum** (should be identical in both versions)
- **Observations about thread performance**, including any bottlenecks or uneven workload distribution
- **Comparison of performance** between the multi-threaded and single-threaded implementations:
    - How much faster (or slower) is the parallel version?
    - Are there diminishing returns or overhead introduced by threading?
    - What factors affect the scalability of your solution?
- **Discussion of scalability**:
    - How might performance change with different input sizes?
    - What would happen if you increased or decreased the number of threads?

You may include tables, charts, or graphs to support your analysis.

4. Thread Design and Load Balancing Explain how you ensured that each thread received a fair share of the work. Consider:

- How you partitioned the input range
- Whether you used static or dynamic scheduling
- Any challenges you encountered in balancing execution time across threads

5. Prime-Finding Algorithm Describe the algorithm you used to identify prime numbers. Address:

- Why you chose this algorithm
- Its time complexity
- Any optimizations you implemented (e.g., skipping even numbers, using a sieve)

6. Reflection Reflect briefly on your experience completing this assignment. Consider:

- What you learned about parallel programming
- What you might improve in a future version
- Whether you used any tools (e.g., AI assistants) and how you verified their output

Submission Instructions

- Submit your written analysis as a separate file via **Webcourses**.
- Be sure your name is included on the document.