[231P] HW3 Prime Numbers

Prarthana Majalikar, 33598574 | Ishan Dhanwal, 88339374 | Ish Kataria, 18183388

1. **System description:**
    - Processor : Apple M3 Pro
    - RAM: 16gb
    - Operative System : MacOS
2. **Comparative table showing execution times:**

Single/Multi Threaded	Number of Threads	Execution time (in secs)
Single	1	0.95821200
Multi	1	0.97384100
Multi	2	0.57940900
Multi	4	0.49037800
Multi	8	0.69197400
Multi	16	1.01291700
Multi	32	1.01808700
Multi	64	1.01833400

3. **Analysis of the performance:**
    - **Correctness:**
        - **Single Threaded:**
            - The single-threaded version uses a simple loop from 1 to 1,000,000 and checks each number if it is prime or not. It's inherently safe and correct because there's no concurrent access to shared data. Output is reliable and consistent.
        - **Multi Threaded:**
            - The multi-threaded version distributes work dynamically using a shared counter protected by a

mutex. Only one thread at a time reads and increments the counter, which ensures no duplicate or skipped numbers. A second mutex guards printf() to ensure output isn't garbled. The dynamic approach prevents idle threads and balances the load better than static division.

- **Performance and Speedup Analysis:**

Threads	Time (s)	Speedup (vs single thread)	Notes
Single Thread	**0.958**	1.00×	Baseline
Multi Thread 1	0.974	0.98×	Slightly slower due to thread overhead
Multi Thread 2	0.579	**1.65×**	Good speedup
Multi Thread 4	0.490	**1.96×**	Best speedup (almost 2×)
Multi Thread 8	0.692	1.38×	Slower than 4 threads
Multi Thread 16	1.012	0.95×	Overhead outweighs benefit
Multi Thread 32	1.018	0.94×	No improvement
Multi Thread 64	1.018	0.94×	Bottleneck due to contention

4. Analysis of single threaded version vs multi threaded with one thread:
    - Single threaded version takes 0.958s and multi threaded version takes 0.974s which is slightly slower than single threaded version. The reason behind would be that the multi threaded version requires thread creation, thread management, and synchronisation using mutexes. The single thread version does not do any mutex locking while the multi threaded version does locking twice, once for counter_mutex to assign number and second print mutex to print safely. This introduces latency even when there is just one thread. Also multi threaded version involves complex/deeper calls to functions like pthread_create() and pthread_join()