Assignment 3 Report
Name: Miguelangel Tamargo
Panther ID: 5866999

In this assignment, I developed a parallel processing application in C that demonstrates the principles of concurrency with multiple threads. The application consists of two threads, each responsible for updating a shared counter. The first thread increments the counter by one and, upon reaching every hundred increments, adds a bonus to its total. The second thread simply increments the counter by one. The application employs mutex locks to manage access to the shared data structure, ensuring that only one thread can modify the counter at any given time. This design effectively showcases a solution to the critical section problem, which is a fundamental issue in concurrent programming.

Various sections of the code are marked with comments related to thread execution, highlighting the entry, critical, exit, and remainder sections for both threads. In the entry section, threads attempt to acquire the mutex lock via the pthread_mutex_trylock() function, ensuring mutual exclusion. This prevents concurrent access to the critical section by allowing only one thread to modify the shared data counter and increment its update count at a time. The critical section is crucial as it safely updates the shared data. Following this, the exit section releases the mutex using pthread_mutex_unlock(), ensuring other threads can attempt to acquire the lock. Progress is maintained as threads continuously attempt to acquire the mutex lock without unnecessary delays, facilitated by the non-blocking nature of pthread_mutex_trylock(), allowing smooth transitions to the critical section when it becomes free. Finally, the remainder section involves printing the results after the loop completes, showing how many times each thread updated the counter. Bounded waiting is respected by the design of pthread_mutex_trylock(), which guarantees threads won't be indefinitely postponed in their attempts to access the critical section. This design allows each thread a fair opportunity to update the shared counter, thereby adhering to all three conditions: mutual exclusion, progress, and bounded waiting.

This assignment provided an interesting and fun opportunity to engage with threading and concurrency concepts in C. The process of managing shared resources while ensuring safe access through mutexes was both challenging and enlightening. I was able to improve my understanding of critical section solutions and effectively implement a multi-threaded application that operates without data corruption. Through the exercise, I now have a better comprehension of parallel processing, and the complexities associated with it, enhancing my programming skills and confidence in managing concurrent applications. C is a very fun and challenging programing language mix. It's very close to the hardware and forces you to understand fundamental concepts that are implemented in everyday higher-level languages that are easily ignored. Making me a more confidant end efficient developer.