Luis Ruiz

CS 218 -1001

Assignment 12

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sequential Timed Executions

-------------------

Timed Test #1

real 0m0.003s

user 0m0.000s

sys 0m0.000s

-------------------

Timed Test #2

real 0m0.002s

user 0m0.000s

sys 0m0.000s

-------------------

Timed Test #3

real 0m0.002s

user 0m0.000s

sys 0m0.000s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Parallel Timed Executions

-------------------

Timed Test #1

real 0m0.001s

user 0m0.000s

sys 0m0.000s

-------------------

Timed Test #2

real 0m0.001s

user 0m0.000s

sys 0m0.000s

-------------------

Timed Test #3

real 0m0.001s

user 0m0.000s

sys 0m0.000s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What I encountered during this assignment is that when my LIMIT was set to a smaller value both the sequential and threaded operations would give me the same correct result at the end. However, when the Limit value was relatively big the threaded operations would have the wrong value, this was because of a race condition that was encountered; my Value was getting overwritten either with the same value that was once placed by one of the threads. We could have prevented this situation if we would have locked the global variable when it was being used and unlocked it when it was available for use.