CS5323 – Operating Systems II
Programming Assignment 1
Due: February 25, 2022, 11:59 p.m.
Submission: via Canvas

In this assignment, you will build upon your code from Assignment 1 to compute global histograms using multiprogramming. Specifically, you will be implementing a shared memory solution to synchronize between two threads. This involves designing a solution to the critical section problem.

Your code must be able to read a file containing several lines of characters. You must use your solution from Assignment 1 to compute the histograms for each line. Instead of printing them for each line, your code must have a global, shared memory in the form of a data structure to record the overall histogram for each character in the *entire file*.

Implement a solution to the critical section problem for N threads using mutex locks. Specifically, in `pthreads` using `pthread` mutex `trylock`. You will need to use `pthread` Unix thread calls to start up N threads from the main process, where N is a command line parameter that can range from 1 to 4. Remember that all global memory is shared among threads of a process.

You will need to look at the `pthread` create, `pthread` join and threads manual pages.

A tutorial and the man pages are here:
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

Information on trylock is here:
http://man.yolinux.com/cgi-bin/man2html?cgi_command=pthread_mutex_lock

Deliverables:

1. A short report describing your implementation details on how you designed a solution to the critical section problem. This should include explanation in terms of the three conditions necessary for a solution to the CS problem. [10 points]
2. A well-documented code implementing the multiprogramming approach to creating global histograms as described above. This includes comments, README file, instructions on compiling and running your code, etc. [40 points]

Things to remember:

1. Submit your assignment as a ZIP file. Include a README with instructions on how to run and expected output along with your report.
2. Each line must be processed by exactly one thread!
3. The result of the global histogram in the example file is given below:
   ```
   {'a': 44, 'b': 24, 'c': 23, 'd': 30, 'e': 63, 'f': 22, 'g': 23, 'h': 25, 'i':
   56, 'j': 23, 'k': 22, 'l': 26, 'm': 22, 'n': 28, 'o': 50, 'p': 25, 'q': 22,
   'r': 40, 's': 25, 't': 36, 'u': 38, 'v': 22, 'w': 22, 'x': 22, 'y': 27, 'z':
   23}
   ```
   Note that this output must be obtained by your code even when running for multiple iterations on CSX to show that it is indeed a solution to the critical section problem.