Run time:
part A:0.019616, for part A the program only has one main thread.

Part B:0.145085, for part B the the program has four threads and they work on different part of the rawData. In order to avoid race conditions, we use mutex; although multiple threads are used to conduct the program, the run time is still longer than part A. Because when we use mutex to avoid race condition, it means the program lock the global array when one thread is modifying it, and other threads need to wait for their turns. So that the run time of part B is longer.

Part C:0.031173, instead of modifying the same global histogram array, part C creates local arrays. Although part C did not use mutex and has multiple threads; however, since the threads have to join together after then terminated, and there are only four threads modifying not that large amount of data, it is still faster to have only one main thread, therefore the run time of part C is still longer then part A, but shorter then part B.

part D: 0.020539, for part D the main creates thread 0 and thread 1 together, create thread 2 and thread 3 together. Later on, the main joins four threads together, and initial two arrays of type int. The function adds the calculated histogram of thread 0 and thread1, then stores the it into array1. Also, the function adds the calculated histogram of thread 2and thread3, then stores in array2. Eventually, adds up the array1 and array2 then stores the answer into finalarray.