**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
CSC 457  
Assignment 5 – Threads**50 points

**Assigned**: 11/9/2022  
**Due:** 11/18/2022

**Evaluation:**

1. Program runs without crashing (\_\_\_\_\_ / 10)
2. Threads are created correctly (\_\_\_\_\_ / 20)
3. Threads (individually) do what they are supposed to do (\_\_\_\_\_ / 20)

***Note: poor indentation, commenting, etc. will result in a lower grade.***

**ASSIGNMENT**:

* Complete this assignment in C. Upload a C source file to Canvas. Your program must be able to run in the VS Code configuration we have used in class.
* You may work in pairs or alone. If you work with a partner, be sure both names are on the work. Only one student from a pair needs to submit the work.
* Write a program with eight threads in addition to the main thread. First, create four threads to produce data. After those threads complete their work, create four other threads that take the produced data and process it. After those threads complete their work, use the main thread to display the results.   
  + STEP 1. The four data-producing threads:
    - All four threads generate random numbers using a random number generator.
    - All of these numbers should be in the range 1 to 1,000,000,000. Even numbers are ignored.
    - Each thread uses its own separate buffer (global variable array) and stores the results in the buffer.
    - Each thread should end after adding 1,000,000 random odd numbers to its global array.
  + STEP 2. The four threads that process the data:
    - Each thread filters for adjacent numbers that are equal. Whenever a number is equal to the following number, pass it on to the next step in another global array. This additional global array is shared by all 4 threads. Each thread processes one of the buffers of 10,000,000 numbers.
  + STEP 3. The main thread displays the data:
    - Display the numbers that made it through the filters from step two.

Note that no synchronization is needed for the first four arrays, since only one thread at a time will access them. The last array, however, is shared by five different threads (the four data processing threads and the main thread). Access to that array must be coordinated.