The purpose of this project is to provide experience with Pthreads and (thread) synchronization.

**Problem Statement**
Complete the producer-consumer project described on pages 274-278 in the textbook using Pthreads. In this project, there may be multiple producers and consumers accessing a bounded buffer, each of which is running in a thread. Differences from the producer-consumer processes with shared memory and semaphores example given in class include:

- Use of threads rather than processes, multiple producers and consumers
- No external shared memory, since threads share global data.
- Use of thread semaphores instead of external semaphores as explained in the project.
- Use of thread mutex lock instead of a mutex semaphore as explained in the project.

In addition, the following requirements must be met:

- There should be output similar to the process synchronization examples shown in class that indicates when a thread is attempting to gain a semaphore and when it releases a semaphore.
- Both the producer and the consumer threads should be passed the loop index number when it is created so that the output indicates which thread produced or consumed an item. I.e., the output of the project should say something like "Producer <#> produced <value>" and "Consumer <#> consumed <value>". Note that the parameter is a char*, so you'll have to convert an int to a char*. The easiest way to do this is to use sprintf() in C or an ostringstream in C++. Use atoi() to convert a char* to an int.

**Assignment**

(20 points) Implementation. This project may be done in individually or in pairs (group of 2). This project may be written in any language as long as it uses Pthreads and runs on cserver. Provide a makefile that will make your project, if needed. Recall that when compiling a Pthread program using gcc or g++ that you must include the command-line option: `-lpthread`

(10 points) Provide a high-level functional analysis and design of the program describing the functionality of the major components of the program and how they interact with each other. If the program does not meet all of the project requirements, describe which parts are missing and what you think should be done to implement them.

(10 points) In addition, answer the following questions:

1. What aspect of thread manipulation did you find most difficult to understand?
2. What aspect of thread manipulation did you find least difficult to understand?
3. What aspect of thread synchronization did you find most difficult to understand?
4. What aspect of thread synchronization did you find least difficult to understand?
5. What, if anything, would you change in your current design?
6. What, if anything, did you find interesting or surprising about thread manipulation or thread synchronization that you did not know before doing this project?

**What to Submit**
Submit items no later than 4:30pm on the due date.

Create a tarfile containing the source code and makefile for the project. Submit this tarfile electronically by emailing it as an attachment to the instructor.

Submit the following items in hardcopy:

- A printout of **well-documented** code for your project, preferably 2-up in landscape mode.
- The functional analysis and design of your project
- Answers to the questions above