# United International University (UIU)

Department of CSE

Trimester: Summer 2022

Course Name: | CSI 310 | Operating Systems Laboratory

## Submission Guideline:

- Put all your java files and output text file inside a folder
- Rename the folder according to your 9 digit student ID
- ZIP the folder and submit it in elms. The name of the zipped folder should also be your 9 digit student ID

Please do not copy codes from others/the internet. Each of the offline assignments will be evaluated. You must be able to explain your code. Also, we will run a copy checker on the submissions. Any plagiarism will be severely penalized.

### Simulate the Producer Consumer Problem

**You must use the template provided in the class.** You can add as your need, but cannot change anything that is already given.

**Required outputs** :

- The template contains instructions about what your output should consist of. You can also look at the sample output.txt file provided.

Prepare a text file (output.txt) with your output lines. You can do this by simply copying the console output to a text file.

Try with multiple producer, consumer, different duration and size to check your code.

At the end of the simulation, print the number of items that was produced, consumed and are still remaining in the buffer.

Theoretical Description of the Problem is given in the next page

## Short Theoretical Description of the Producer Consumer Problem:

In computing, the producer–consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer (a queue) which will be set by user. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time.

The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer. The solution for the producer is to go to sleep if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer. The solution can be reached by means of inter-process communication, typically using semaphores. An inadequate solution could result in a deadlock where both processes are waiting to be awakened. The problem can also be generalized to have multiple producers and consumers.