**Concordia University**

**comp346 - Summer 2020**

**Operating Systems**

**Programming assignment 3**

**Deadline:** By 11:59pm, Friday Aug. 7 2020

**Late Submission:** No late submission

**Teams:** The assignment can be done individually or in teams of 2 or 3. Submit only one assignment per team.

**Purpose:** The purpose of this assignment is to code the bounded buffer problem using Semaphore.

* **Problem specification.**

In this assignment, you will implement the bounded buffer synchronization solution using semaphore. You need to implement your own Semaphore class with its two functions: wait() and signal(). Then you need to have a producer class and a consumer class. When your program runs it should, first, ask the user to set a value for q (defined latter in the implementation section). The program should display a trace of what is happening as far as:

Generation of an items

Consumption of an item

The state of the buffer (full or empty)

The value of all used semaphores

The state of the two threads and whether any of them is in the state of busy waiting

* **Implementation.**

The producer produces items with a probability rate q and the consumer with a probability rate 1-q where q ranges between 0 and 1. When q is set to 0, that means no items are produced.

When q is set to 1, that means no items are consumed (only items are produced).

When q is set to 0.5 that means there is an equal probability of producing an item or consuming an item.

When q is set to 6.66, that means the likelihood of producing an item is twice as high as consuming one. So eventually, the buffer will get full.

You should have a loop, in every iteration, you generate two random numbers between 0 and 1: P and C. If P is < q, the producer will try to produce an item. If C < 1-q, the consumer will try to consume an item.

Use an array for the buffer. The size of the array should be defined as a constant. In your demo, you can set that constant to 10.

* **Evaluation.**

You will be evaluated mostly on the implementation of the required functionality, the implementation of the algorithms, and the correctness of the output.

***If you miss your demo time, you will receive 0 credit for the assignment.***

***Evaluation Criteria***

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| The semaphore class | 25% |
| Proper consumption of items | 20% |
| Proper production of items | 20% |
| Required info in the output | 20% |
| The handling of q | 15% |

**Required documents.**

* Source code in Java

**Submission.**

Create one zip file, containing the necessary files (.java, .txt and test cases). If the assignment is done individually, your file should be called pa1\_studentID,where pa1 is the number of the assignment and studentID is your student IDnumber. If the work is done in a team of 2 or 3 people, the zip file should be called pa1\_studentID1\_studentID2 or pa1\_studentID1\_studentID2\_studentID3 where studentID1, studentID2, and studentID3 are the studentID numbers of each student.

The zip file should be uploaded to Moodle before the due date. No late submissions are accepted.