WEEKS 8 & 9: Producer Consumer Problem Date: 27/10/2020

OBJECTIVE:

PROGRAM 1: Write a C program to implement Producer Consumer problem using Semaphores

PROGRAM 2: Write a C program to implement Producer Consumer problem using Pipes

- **CONCEPTS ARE ALREADY COVERED IN THEORY**
- **STUDENTS ARE ADVISED TO REFER TO THE TEXT BOOK, LECTURE MATERIAL AND PROGRAMS DEMONSTRATED IN THE CLASS TO IMPLEMENT THE GIVEN PROGRAMS.**
- **STUDENTS ARE REQUIRED TO PROVIDE PROOF OF CONDUCTION (AS PER SUBMISSION PROCEDURE BELOW).**

SUBMISSION:

1. **The source code files for both PROGRAM 1 and PROGRAM 2 should be uploaded to EDMODO in WORD or ZIP FORMAT.**
2. **All the screenshots clearly showing the directory name as YOURSRN_NAME_WEEK8-9 for both the programs should be uploaded to EDMODO in a SINGLE FILE (Word or PDF format only, Do NOT zip this file).**

Students should keep these TWO deliverables (i.e. 1 & 2 above) separate and NOT zip all the files together in order to facilitate quick, timely and effective evaluation.

Contact your respective Lab faculty for any questions or clarifications needed.

DUE DATE FOR SUBMISSION: IN VIEW OF THE HOLIDAYS, SUBMIT BOTH PROGRAM 1 and PROGRAM 2 TOGETHER ON OR BEFORE 05/11/2020 11:59 PM

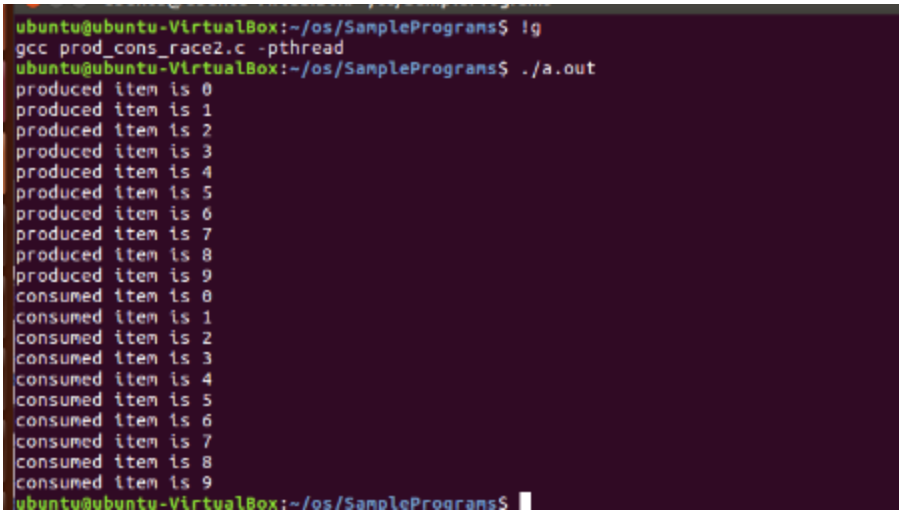
PROGRAMS FOR EXECUTION AND SUBMISSION:

PROGRAM 1: Write a C Program to implement Producer-Consumer Problem using Semaphores

This is similar to Week 7 Program but using Semaphores instead of Mutex locks. Implement a main program that creates two threads: producer and consumer threads which execute producer and consumer functions respectively. The producer should produce an item and update the buffer. The consumer should consume an item and update the buffer. Use Semaphores (preferably **unnamed** semaphores) to enclose the critical sections in both producer and consumer so that only one of them can update the buffer at a time and **prevent race condition** as shown in the sample output below. Note: consumer should **wait** if buffer is empty and producer should **signal** when the buffer has at least one item. You can use bounded buffer.

Sample Output:

Your output can vary depending upon your implementation and the manner in which the threads are scheduled on your system but you should not see race condition.



PROGRAM 2: Write a C program to implement Producer Consumer problem using Pipes.

Create a simple pipe between producer and consumer using pipe() system call. The producer should write information to the write end of the pipe and the consumer should read information from the read end of the pipe and copy it to stdout.

Sample Output:
Your output can slightly vary depending upon your implementation

```
ubuntu@ubuntu-VirtualBox:~/os/SamplePrograms$ ./a.out
Producer sent: 1
Producer sent: 2
Producer sent: 3
Producer sent: 4
Producer sent: 5
Producer sent: 6
Producer sent: 7
Producer sent: 8
Producer sent: 9
Producer sent: 10
consumer: got 1
consumer: got 2
consumer: got 3
consumer: got 4
consumer: got 5
consumer: got 6
consumer: got 7
consumer: got 8
consumer: got 9
consumer: got 10
ubuntu@ubuntu-VirtualBox:~/os/SamplePrograms$
```