CSE 2431 **LAB 1** **AUTUMN 2018**

Due: Tuesday, September 18, 2018 at 11:59 p.m.
5 points

This lab helps you understand the concept of processes, practice with using fork() and exec() to create new processes, and perform shared memory based inter-process communication.

The **source code** and **Makefile** must be packed as a single zip file and submitted electronically online in Carmen Dropbox by **11:59 pm Tuesday 09/18/18. Late penalty will be 25% per day.** Please complete your solution in the virtual machine you installed in Lab 0. The grader will compile and test your solution under the same environment. Any program that does not compile will receive a zero. The grader will NOT spend any time fixing your code. It is your responsibility to leave yourself enough time to ensure that your code can be compiled, run, and tested well before the due date.

Although it is fine to talk with other students at a high-level regarding concepts related to the lab, the lab must be the student's own individual work. You may use Carmen to communicate with other students in the class. Guidelines for Carmen and communication are given on the Syllabus.

Questions: 5 points
In this lab, you are asked to implement a bounded-buffer producer-consumer communication using POSIX shared memory on Linux. You will need to complete two programs: producer.c and consumer.c. They both include a header file, headers.h. The header file and the Makefile have been created for you, and you don't need to modify them.

You will launch the first process, *producer*, from the command line. The *producer* will fork() a child process which will use exec() to execute *consumer*. Then the *producer* and the *consumer* will create a bounded buffer on top of the POSIX shared memory.

The *producer* will read from a file *input.txt* and convert each line of the file into an element of the bounded buffer and send the element to the *consumer*. The *consumer*, upon receiving the message, will write the content to output.txt one line after another. The producer sends an empty element with id = -1 to indicate the end of the messages. The consumer will terminate once an empty element is received.

Most of the framework of producer.c and consumer.c have been created for you already. You are expected to complete the code and test it by yourself.

Extra credit: 1 point
The above experiment has one producer and one consumer. Can you implement a version with one producer and two consumers? Would you still be able to make sure output.txt is the same as input.txt? What if your virtual machine has multiple virtual CPUs? Write a lab report to summarize your findings, and discuss possible solutions.