University of Engineering and Technology, Lahore
Operating Systems Lab.

Semester: 6th **Session: 2019-23**

Lab-09 (Individual)

Submission Date: -04-22 (Before 12:00 pm)
Note: Understanding the assignment is part of the assignment.

Lab Topic:

-Semaphore through POSIX API

Lab Objectives:

- Understanding critical section, race condition and synchronization

Inlab Questions

Task 1:

There are exactly 3 threads generate string a, b and c in an arbitrary order. In an absence of any synchronization mechanism there will be no order in generation of a, b and c. In the form of regular expression the string (a | b | c)* {* means many times a character an occur, | means or, so different combinations can be aaaaaaa… , bbbbbbbbbbb… , ccccc… }. Synchronize threads using semaphore in such a way that your printed string will be (cba)* {* means many times cba can occur, so different combinations will be cbacbacbacb….}.

Note you are not allowed to add or delete any cout statement

//thread 1
While(1)
{
Cout << ‘a’;
}
	//thread 1
	While(1)
	{
	Cout << ‘b’;
	}
		//thread 1
		While(1)
		{
		Cout << ‘c’;
		}

Modify task 1 so that printed string will be (cbaaadd)*

Task 2:

AIM:

In this assignment you will be practicing semaphores for process synchronization.

ALGORITHM:

1. In your main program create two threads. Name one as producer and other as consumer.

2. Create a buffer of size 10 in your main program.

3. Now after every 2 seconds producer should produce something and consumer should consume it. Display message on screen when an item is produced and consumed. You can use random function to produce an item.

4. This procedure will run indefinitely. Synchronize your producer and consumer process using semaphores.