National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

(CL-2006)

Course Instructor	Ms. Rubab Anam
Lab Instructor	Muhammad Faheem
Section	BCS 4F
Semester	Spring 2025

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives:

- In this lab, students will practice thread and process synchronization using semaphores.

Lab Questions

Question 1(3):

There are exactly 3 threads that generate string a, b, and c in an arbitrary order. In the absence of any synchronization mechanism, there will be no order in the generation of a, b, and c. Get cli arguments denoting the number of times a, b and c will print respectively in a single substring and the order from cli will determine their printing order, also the last argument is for the number of times the substring should be printed. Thread 1 should print a, thread 2 should print b and thread 3 should print c.

Eg If cli is **./q1 a 3 c 1 b 1 3** Synchronize threads using semaphore in such a way that your printed string will be aaacbaaacbaaacb…… Here a substring is formed with thread 1 printing a 3 times, thread 3 prints c and then thread 2 prints b. Last argument 3 denotes that the substring formed should be printed 3 times.

```
//thread 1
While(1)
{
 cout << 'a';
}
```

```
//thread 2
While(1)
{
 Cout << 'b';
}
```

```
//thread 3
While(1)
{
 Cout << 'c';
}
```

Question 2(3):

Two Threads i.e., Producer and Consumer are sharing a buffer of size 100. The producer generates a random number from 1 to 6, stores that number in the buffer, and prints the number. The consumer must read values from the buffer added by the producer. If the producer generates Number 6, the consumer should perform the cumulative summation operation starting between successive 6's, and display the result. The consumer should also print the sum after 10 non six values but the cumulative sum should only go to 0 after a 6. The producer will stop after adding 100 values. Both producer and consumer threads run simultaneously; hence, synchronize.

You're not allowed to use cin/scanf statements.

The output should look Like

```
Number 2 //printed by producer.
Number 5
Number 4
Number 2
```

Number 6
Sum 13 //printed by consumer
Number 2
Number 5
Number 1
Number 4
Number 2
Number 3
Number 4
Number 3
Number 5
Number 5
Sum 34//printed by consumer
Number 6
Sum 34 //printed by consumer.

Question 3(4):

Suppose there are 2 **unrelated** processes. Each process reads a different file having a list of integers. Both processes then read the integers, calculate their sum, and send the sum and the count of integers to a server process via shared memory. The server then finds the total average by following formulae:

Total Sum= sum of integers sent by p1 + sum of integers sent by p2

Total Count= count sent by p1 + count sent by p2

Average= Total Sum/ Total Count

After calculating the average, the server sends the average to both processes. Both processes then print the sum on their respective terminals. (You need to synchronize the processes using semaphores on shared memory)

Shared Memory Portions Required:

1. There will be a single shared memory portion on which both processes will place their (sum, count) pair. One process will put the pair on the 0^{th}index and the other will put the pair on the 1^{st}index.

2. A shared memory portion for the currently available index number where a process can put the pair.

Notes:

Read man pages of sem_overview and shm_overview for semaphores and shared memory.
System calls: sem_wait, sem_post, sem_init, sem_destroy etc.