National University of Computer and Emerging Sciences
Lahore Campus

Operating Systems (CS2006)

Date: April 4st 2024

Course Instructor(s)

Mr. Razi Uddin
Mr. Mubashar Hussain
Ms. Rubab Anam
Ms. Namra Absar

Sessional-II Exam

Total Time (Hrs):	1
Total Marks:	38
Total Questions:	3

Roll No _____ Section _____ Student Signature _____

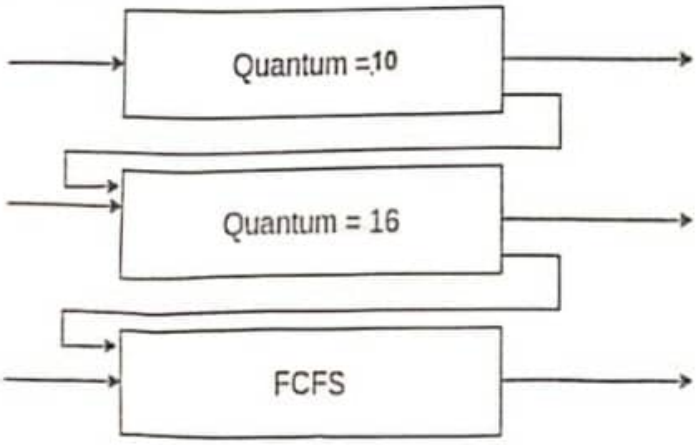
Do not write below this line

Attempt all the questions.

Q1: [CLO-3] **[Marks: 15]**

Consider a multilevel feedback queue scheduling with three queues, numbered as Q1,Q2,Q3.

- The scheduler first executes processes in Q1, which is given a time quantum of 10 milli-seconds.
- If a process does not finish within this time, it is moved to the tail of the Q2.
- The scheduler executes processes in Q2 only when Q1 is empty.
- The process at the head of the Q2 is given a quantum of 16 milli-seconds.
- If it does not complete, it is preempted and is put into Q3.
- Processes in Q3 are run on an FCFS basis, only when Q1 and Q2 are empty.
- A process that arrives for queue 2 will preempt a process in queue 3. A process in queue 2 will in turn be preempted by a process arriving for queue 1.
- If a process does not use up its quantum in queue 2 due to preemption by queue 1, it will keep its current queuing level and be put into the end of the queue. Then, it can still get the same amount of quantum (not remaining quantum) next time when it is picked.



The following set of processes, with the arrival times and the length of the CPU-burst times given in milliseconds, have to be scheduled using this Multilevel Feedback Queue Scheduler:

Processes	Arrival time	Burst time
P1	0	17
P2	12	25
P3	28	8
P4	36	32
P5	46	18

a) Draw a Gantt chart illustrating the execution of these processes.
b) Calculate the average waiting time and the average turnaround time for the scheduling.

Q2: [CLO-2] **[Marks: 8]**

A process has 3 threads, T1, T2, and T3, and its code does not contain any exec*() commands. T1 opens 3 files and T2 creates a pipe. After these actions, T1 executes a fork() command and T2 executes a fork() whose child immediately calls execvp() to execute a single-threaded program. T1 closes the three files and then terminates. Note that all processes use Pthreads.

a. How many different processes are running? Why? **[Marks: 2]**
b. How many different programs are being executed? Why? **[Marks: 2]**
c. How many open file descriptors are there? Why? **[Marks: 3]**
d. Thread T3 makes a system call that causes all threads in its process to terminate. Which system call could it be? **[Marks: 1]**

Q3: [CLO-3] **[Marks: 10+5=15]**

A) Suppose we want to synchronize two concurrent processes P and Q using binary semaphores S and T. The code for the processes P and Q is shown below. Synchronize them using Semaphores so that it always lead to an output string with **"000110001100011"**.

Process P	Process Q

while(1)
{
 	
cout<<"0";
 	
}
	while(1)
	{
	
	cout<<"1";
	
	}

B) Consider multi-threaded system in which two threads running this fragment of code simultaneously, s1, s2, s3 and s4 are shared semaphores, all are initiliaze to 1. While all other variables are automatic, that is each thread has a local copy of a and b that it modifies, initial values are a=2 and b=0 in Thread1 and a=0 and b=2 in Thread2.

P (or **wait**) is used to acquire a resource.
V (or **signal**) is used to release a resource.

Consider the following program fragment:

```
if(a > 0)
        P(s1);
else
        P(s2);
b++;
P(s3);
if(b < 0 && a <= 0)
        P(s1);
else if(b >= 0 && a > 0)
        P(s2);
else
        P(s4);
a++;
V(s4);
V(s3);
V(s2);
V(s1);
```

1. After running this fragment of code simultaneously, can there be a deadlock? Why, or why not?
2. In above case, state all the possible values of a, b and shared semaphores to justify part 1, for both threads

BEST OF LUCK!