OPERATING SYSTEM PRACTICE QUESTIONS

1. A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the processes W and X reads x from memory, increments by one, stores it to memory, and then terminates. Each of the processes Y and Z reads x from memory, decrements by two, stores it to memory, and then terminates. Each process before reading x invokes the P operation (i.e., wait) on a counting semaphore S and invokes the V operation (i.e., signal) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution?
2. Consider a uniprocessor system executing three tasks T1, T2 and T3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T1, T2 and T3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1st milliseconds and task preemptions are allowed, In how many milliseconds the first instance of T3 completes its execution?
3. Let us consider a method used by set of two concurrent processes P1 and P2 in a uniprocessor system for accessing some shared resource. Two shared Boolean variable S1 and S2 used by process P1 and P2 to synchronize their activities and initial value of S1 and S2 is randomly assigned. Methods are:

P1	P2
While(S1==S2);	While(S1!=S2);
Critical Section	Critical Section
S1=S2	S1= not S2

   Is the method ensuring synchronization between the process P1 and P2? Justify.

4. You are tasked with simulating the Dining Philosophers problem, where five philosophers sit around a circular table, and they alternate between thinking and eating. To eat, a philosopher must pick up two forks placed between them and their neighbors. Describe the challenges and potential solutions to this problem using Semaphore.

   If we add a 6th chopstick to the center of the table, have we cured the deadlock problem?

If yes, what condition have we removed? If no, explain why not.

5. Consider three CPU-bound processes arriving to a First Come First Served (FCFS) scheduler as follows:

Process	**Arrival time**	**Burst time**
P1	0	5
P2	1	12
P3	5	3
P4	6	4

Calculate the average waiting time across all FOUR processes for Round Robin CPU Scheduling Algorithm.

6. Give two reasons why caches are useful. What problems do they solve?
7. What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?
8. Three processes X, Y and Z each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires tc CPU milliseconds and then initiates a single I/O operation that lasts for tio milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also, the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process id	tc	tio
X	100 ms	500 ms
Y	350 ms	500 ms
Z	250 ms	500 ms

The processes X, Y, and Z are started at times 0, 5 and 10 milliseconds respectively, in a pure time-sharing system (round robin scheduling) that uses a time slice of 50 milliseconds. Calculate the time in milliseconds at which process Z would complete its first I/O operation.

9. For the processes listed in the following table, which of the following scheduling schemes will give the lowest average turnaround time?

Process	Arrival Time	Processing Time
A	0	3
B	1	6
C	4	4
D	6	2

A. First Come First Serve
B. Non-preemptive Shortest job first
C. Shortest Remaining Time
D. Round Robin with Quantum value two