Operating Systems and Systems Programming (15B11CI412)

Tutorial - 5

Process Synchronization

Q1. Define the meaning of a race condition? Use an execution sequence to illustrate your answer.

Q2. A good solution to the critical section problem must satisfy three conditions: mutual exclusion, progress and bounded waiting. Explain the meaning of the progress condition. Does starvation violate the progress condition?

Q3. Discuss the tradeoff between fairness and throughput of operations in the readers-writers problem. Propose a method for solving the readers-writers problem without causing starvation.

Q4. What is the meaning of the term busy waiting? What other kinds of waiting are there in an operating system? Can busy waiting be avoided altogether? Explain your answer.

Q5. Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in user level programs.

Q6. Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems.

Q7. Demonstrate that monitors and semaphores are equivalent in so far as they can be used to implement the same types of synchronization problems

Q8. Write a monitor that implements an *alarm clock* that enables a calling program to delay itself for a specified number of time units (*ticks*). You may assume the existence of a real hardware clock, which invokes a procedure *tick* in your monitor at regular intervals.

Q9. Consider three concurrently executing threads in the same process using two semaphores s1 and s2. Assume s1 has been initialized to 1, while s2 has been initialized to 0.
What are the possible values of the global variable x, initialized to 0, after all three threads have terminated?

```
/* thread A */
P(&s2);
P(&s1);
x = x*2;
V(&s1);
/* thread B */
P(&s1);
```

```
x = x*x;
V(&s1);
/* thread C */
P(&s1);
x = x+3;
V(&s2);
V(&s1);
```

Q 10. The following pair of processes share a common variable X:

Process A	Process B
int Y;	int Z;
A1: Y = X*2;	B1: Z = X+1;
A2: X = Y;	B2: X = Z;

X is set to 5 before either process begins execution. As usual, statements within a process are executed sequentially, but statements in process A may execute in any order with respect to statements in process B.

A. How many different values of X are possible after both processes finish executing?

B. Finally, suppose the programs are modified as follows to use a shared binary semaphore T:

Process A	Process B
int Y;	int Z;
A1: Y = X*2;	B1: wait(T);
A2: X = Y;	B2: Z = X+1;
signal(T);	X = Z;

T is set to 0 before either process begins execution and, as before, X is set to 5. Now, how many different values of X are possible after both processes finish executing?