King Saud University
College of Computer & Information Sciences
Department of Computer Science

CS227: Operating Systems **Tutorial 6** **Second Semester 1442H**

Q1) Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes:

```
/* P1 */
while (true) {
  wants1 = true;
  while (wants2 == true);
  /* Critical  Section */
  wants1 = false;
}
/* Remainder section */
```

```
/* P2 */
while (true) {
  wants2 = true;
  while (wants1 == true);
  /* Critical Section */
  wants2 = false;
}
/* Remainder section */
```

Answer: d
Solution:
Now, when both wants1 and wants2 become true, both process p1 and p2 enter in while loop and waiting for each other to finish. This while loop run indefinitely which leads to deadlock.
Now, Assume P1 is in critical section (it means wants1=true, wants2 can be anything, true or false). So this ensures that p2 won't enter in critical section and vice versa. This satisfies the property of mutual exclusion.
Here bounded waiting condition is also satisfied as there is a bound on the number of process which gets access to critical section after a process request access to it.

Q2) Consider the two cooperative processes below. You are required to identify the critical sections (CS) between P1& P2 (if any)

Line #	P1
1	message= "hello world"
2	print message
3	x = x + 1
4	sleep(1)
5	print x

Line #	P2
1	y=100
2	msg= "hello world"
3	print msg
4	x = x + y
5	y = x

Q3) The following is a set of three interacting processes that can access two shared semaphores:

```
semaphore U = 3;
semaphore V = 1;

[Process 1]          [Process 2]        [Process 3]

Loop1:wait(U)       Loop2:wait(V)      Loop3:wait(V)
      type("A")             type("B")          type("D")
      signal(V)             type("C")          goto Loop3
      goto Loop1            signal(U)
                            goto Loop2
```

King Saud University
College of Computer & Information Sciences
Department of Computer Science

CS227: Operating Systems **Tutorial 6** **Second Semester 1442H**

Within each process the statements are executed sequentially, but statements from different processes can be interleaved in any order that's consistent with the constraints imposed by the semaphores. When answering the questions below assume that once execution begins, the processes will be allowed to run until all 3 processes are stuck in a wait() statement, at which point execution is halted.

NOTE: Solve it using semaphore queue not busy waiting solution.

A) Assuming execution is eventually halted, how many A's are printed when the set of processes runs?
B) Assuming execution is eventually halted, how many D's are printed when this set of processes runs?
C) What is the smallest number of B's that might be printed when this set of processes runs? 0
D) Is ADBCAABCDD a possible output sequence when this set of processes runs? Show the events that implied by this sequence.
E) Is ABCBADCADBCA a possible output sequence when this set of processes runs? Show the events that implied by this sequence.