**SEMAPHORES**

1. An un-interruptible unit is known as \_\_\_\_\_\_\_\_\_\_\_\_  
   a) single  
   b) atomic  
   c) static  
   d) none of the mentioned
2. TestAndSet instruction is executed \_\_\_\_\_\_\_\_\_\_\_\_  
   a) after a particular process  
   b) periodically  
   c) atomically  
   d) none of the mentioned
3. 3. Semaphore is a/an \_\_\_\_\_\_\_ to solve the critical section problem.  
   a) hardware for a system  
   b) special program for a system  
   c) integer variable  
   d) none of the mentioned
4. 4. What are the two atomic operations permissible on semaphores?  
   a) wait  
   b) stop  
   c) hold  
   d) none of the mentioned
5. What are Spinlocks?  
   a) CPU cycles wasting locks over critical sections of programs  
   b) Locks that avoid time wastage in context switches  
   c) Locks that work better on multiprocessor systems  
   d) All of the mentioned
6. What is the main disadvantage of spinlocks?  
   a) they are not sufficient for many process  
   b) they require busy waiting  
   c) they are unreliable sometimes  
   d) they are too complex for programmers
7. The wait operation of the semaphore basically works on the basic \_\_\_\_\_\_\_ system call.  
   a) stop()  
   b) block()  
   c) hold()  
   d) wait()
8. The signal operation of the semaphore basically works on the basic \_\_\_\_\_\_\_ system call.  
   a) continue()  
   b) wakeup()  
   c) getup()  
   d) start()
9. If the semaphore value is negative \_\_\_\_\_\_\_\_\_\_\_\_  
   a) its magnitude is the number of processes waiting on that semaphore  
   b) it is invalid  
   c) no operation can be further performed on it until the signal operation is performed on it  
   d) none of the mentioned
10. The code that changes the value of the semaphore is \_\_\_\_\_\_\_\_\_\_\_\_  
    a) remainder section code  
    b) non – critical section code  
    c) critical section code  
    d) none of the mentioned
11. The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S0 = 1, S1 = 0, S2 = 0.

Process P0

while(true)

{

wait(S0);

print '0';

release(S1);

release(S2);

}

Process P1

wait(S1);

release(S0);

Process P2

wait(S2);

release(S0);

How many times will P0 print ‘0’?  
a) At least twice  
b) Exactly twice  
c) Exactly thrice  
d) Exactly once

12. Each process Pi, i = 0,1,2,3,……,9 is coded as follows.

repeat

P(mutex)

{Critical Section}

V(mutex)

forever

The code for P10 is identical except that it uses V(mutex) instead of P(mutex). What is the largest number of processes that can be inside the critical section at any moment (the mutex being initialized to 1)?  
a)1  
b)2  
c)3  
d) None of the mentioned

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

Process P1 :

while(true)

{

w1 = true;

while(w2 == true);

Critical section

w1 = false;

}

Remainder Section

Process P2 :

while(true)

{

w2 = true;

while(w1 == true);

Critical section

w2 = false;

}

Remainder Section

Here, w1 and w2 have shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct?  
a) It does not ensure mutual exclusion  
b) It does not ensure bounded waiting  
c) It requires that processes enter the critical section in strict alternation  
d) It does not prevent deadlocks but ensures mutual exclusion