## Synchronization

1) For correct and efficient operation using shared data, a synchronization solution must ensure: **[MSQ]**
A) No two processes can be inside their critical sections at same time.
B) No assumptions needed about relative process speeds or number of CPUs.
C) No process stopped outside its critical section can block other processes.
D) No process can be indefinitely postponed from entering its critical section.

2) _____ is a situation in which two or more processes continuously change their state in response to changes in the other process(es) without doing any useful work, indefinitely.
A) Spin lock
B) Deadlock
C) Livelock
D) None of these.

3) Which of the following statements is/are correct about the Test-And-Set lock (TSL) synchronization mechanism? **[MSQ]**
A) TSL requires hardware support for its implementation.
B) TSL does not guarantee bounded waiting.
C) TSL may suffer from priority inversion problems.
D) TSL may lead to deadlock.

4) Which of the following is a busy wait or uses spin lock for synchronization? **[MSQ]**
A) Test-And-Set lock (TSL)
B) Peterson's solution
C) Compare-and-Swap instruction (CAS)
D) Semaphores

5) Consider the following code for process synchronization using a busy waiting mechanism (spin lock). The system employs Round robin scheduling with a quantum of 1 cpu clock cycle (cc), and each high level instruction takes 1 clock cycle for execution. Assume process P0 is scheduled first and the shared variable turn is initialized to 1. The time taken to complete the execution of both the processes is _____ clock cycles. **[NAT]**

| Process P0 | Process P1 |
|---|---|
| while(turn==1);<br>a=a+b;<br>turn=1; | while(turn==0);<br>a=a+b;<br>turn=0; |

6) Consider the following partially filled table obtained as a result of operation by two processes P1 and P2, on two counting semaphores S1 and S2, both initialized to 0.

| Time | Process | Action | Blocked | S1 | S2 |
|---|---|---|---|---|---|
| 0 | - | - | - | 0 | 0 |
| 1 | P1 | V(S1) | | 1 | 0 |
| 2 | P1 | P(S2) | P1 | 1 | ***X*** |
| 3 | P2 | V(S2) | | 1 | 0 |
| 4 | P2 | ***Y*** | | 0 | 0 |

The correct options for the place holders X and Y are ?
A) X = -1, Y = P(S1)
B) X = 1, Y = P(S2)
C) X = 0, Y = V(S1)
D) X = -1, Y = V(S2)

7) Consider the following synchronization mechanism for Producer/Consumer problem, where,
Producer:

- creates data and adds to the buffer
- do not want to overflow the buffer

Consumer:

- removes data from buffer (consumes it)
- does not want to get ahead of producer

Information common to both processes:
empty := n // where n is the number of buffer slots.
full := 0
mutex := 1

| Producer Process | Consumer Process |
|---|---|
| | |

| | |
|---|---|
| repeat<br>produce an item in nextp<br><br>wait(empty);<br>wait(mutex);<br><br>add nextp to buffer<br><br>signal(mutex);<br>signal(full);<br>until false; | repeat<br><br>wait(full);<br>wait(mutex);<br><br>remove an item from buffer to nextc<br><br>signal(mutex);<br>signal(empty);<br><br>consume the item in nextc<br><br>until false; |

Which of the following are correct about the above mechanism?**[MSQ]**

A) The above mechanism is the correct implementation for the Producer Consumer problem.
B) The mutex semaphore is used to ensure mutually exclusive access to the buffer.
C) The empty and full semaphores are used for process synchronization.
D) If we change the code in the consumer process from: wait(full), wait(mutex) to wait(mutex), wait(full), then a deadlock may occur.

8) Identify the correct statement from the following :

A) The Test-and-Set Lock hardware synchronization mechanism suffers from spin-lock when a round-robin scheduling algorithm is used.
B) Peterson's solution to the mutual-exclusion problem works for non-preemptive scheduling.
C) An operating system that can disable interrupts can implement semaphores.
D) None of these.

9) Consider the following synchronization mechanism using semaphores S1 and S2 :

| PROCESS 1 | PROCESS 2 |
|---|---|
| While (1) {<br>// Do something<br>S1.Wait();<br>Print("1");<br>S2.signal();<br>//Do something<br>} | While (1) {<br>// Do something<br>S2.Wait();<br>Print("2");<br>S1.signal();<br>//Do something<br>} |

The initial value of S1 and S2 so that we can obtain the sequence 1,2,1,2,1,2..... are

A. S1=0, S2=1
B. S1=1, S2=0
C. S1=0, S2=0
D. S1=1, S2=1

10) Select the correct options among the following: **[MSQ]**

A) Starvation-freedom imply deadlock-freedom.
B) Starvation-freedom imply bounded-waiting.
C) Bounded-waiting imply starvation-freedom
D) We need Progress and Bounded-Waiting both to imply Deadlock-Freedom.

11) Consider two processes A and B that uses binary semaphore to implement synchronization mechanism on three resources associated with binary semaphores a,b and c. The process A access resources as follows :
A : P(a); P(b); P(c);

Which of the following access order by process B will not lead to deadlock ?
I) P(a); P(b); P(c);
II) P(a); P(c); P(b);
III) P(c); P(b); P(a);
IV) P(b); P(a); P(c);

A) I only
B) I and II only
C) I,II and III only
D) All I, II , III and IV.

12) Consider the following pseudo-code for a process Pi, where "shared boolean flag[2]" is a variable declared in shared memory, initialized as: flag[0] = flag[1] = FALSE;

```
// i=0 for P0, i=1 for P1
P (int i) {
 while (TRUE) {
   flag[i] = TRUE;
    while (flag[(i+1) % 2) == TRUE);
          < Critical Section >
    flag[i] = FALSE;
 < Remainder Section >
 }
}
```

Which of the following is FALSE?

A) Mutual exclusion is achieved.
B) Progress is achieved.
C) The above code uses the spin-lock mechanism.
D) All of them.

Deadlock implies no progress

13) Consider the following code in which item=0, A and B are shared variables. Pick the correct choice from the given options.

| Process A | Process B |
|---|---|
| 1. int A = 1;<br>2. while(1){<br>3. if (B == 0) {<br>4. if (item == 0) {<br>5. item=item+1;<br>6. }<br>7. }<br>8. A = 0;<br>9. } | 1. int B = 1;<br>2. While (1) {<br>3. if (A == 0) {<br>4. if (item == 0) {<br>5. item=item+1;<br>6. }<br>7. }<br>8. B = 0;<br>9. } |

A) The above code ensures mutual exclusion on shared variable item.
B) The above code does not ensure mutual exclusion on shared variable item.
C) The above code ensures bounded waiting.
D) None of the above.

14) Consider the following code for process synchronization using two binary semaphores S1 and S2, both initialized to 1.

| Process P1 | Process P2 |
|---|---|
| P(S1);<br>P(S2);<br>//Critical Section<br>…… | P(S2);<br>P(S1);<br>//Critical Section<br>…….. |

The probability that the above synchronization code leads to a deadlock is _____. **[NAT]**

15) When a process does not get access to the resource, it loops continually for the resource and wastes CPU cycles. It is known as

(A) deadlock
(B) spinlock
(C) livelock

(D) none

16) The operations that cannot be overlapped or interleaved with execution of any other operations are known as

(A) atomic operations
(B) messages
(C) system calls
(D) none

16) In a _____, the process that locks the Critical Section will only unlock it.

(A) binary semaphore
(B) mutex
(C) counting semaphore
(D) none

17) The following function (foo) is called by multiple processes (potentially concurrently). Identify the critical section(s) that require(s) mutual exclusion.

```
Line 1:   int i;

Line 2:   void foo()
Line 3:   {
Line 4:      int j;

Line 5:      /* Code Section A*/

Line 6:      i = i + 1;
Line 7:      j = j + 1;

Line 8:      /* Code Section B */
Line 9:   }
```

A) Line 6 and Line 7
B) Line 6 only
C) Line 7 only
D) Line 5,6, 7 and 8 .

18) Consider the following synchronization code using binary semaphores:

| semaphore *mutex =1, *data = 1; | |
|---|---|
| Process P1 | Process P2 |
| P(mutex);<br>/* do something */<br><br>P(data);<br>/* do something else */<br><br>V(mutex);<br><br>/* clean up */<br>V(data); | P(data)<br>P(mutex);<br><br>/* do something */<br><br>V(data);<br>V(mutex); |

The above code :
A) Does not ensure mutual exclusion
B) Free from deadlock
C) None of these.
D) The above code is not deadlock free.

19) Consider the following synchronization code using binary semaphores:

| semaphore *mutex =1, *data = 1; | |
|---|---|
| Process P1 | Process P2 |
| P(mutex);<br>/* do something */<br><br>P(data);<br>/* do something else */<br><br>V(mutex);<br><br>/* clean up */<br>V(data); | P(mutex);<br>P(data)<br><br>/* do something */<br><br>V(data);<br>V(mutex); |

The above code :

A) Does not ensure mutual exclusion
B) Free from deadlock
C) None of these
D) The above code is not deadlock free.

20) Consider the following two processes synchronization code

```
shared int turn = 1;
int My_Pid = 0; // 1 for other process.
int Other_Pid = 1 – My_Pid;
while(1){
    while(turn != My_Pid);

    // Critical Section

    turn = Other_Pid;

    // Remainder Section
}
```

Which of the following is TRUE?

A) The above code fails to achieve the progress requirement.
B) The above code fails to achieve Mutual Exclusion.
C) The above code fails to achieve Bounded Waiting.
D) The above code may lead to Deadlock.

21) Consider the following two processes synchronization code :

Because of deadlock progress is not satisfied

```
shared boolean wants[2] = false;
int My_Pid = 0; // 1 for other process.
int Other_Pid = 1 – My_Pid;

while(1){
    wants[My_Pid] = true;
    while(wants[Other_Pid]);

    // Critical Section

    wants[My_Pid] = false;

    // Remainder Section
}
```

Which of the following is/are TRUE? [MSQ]

A) The above code achieves Mutual Exclusion.
B) The above code achieves Progress.
C) The above code employs Spin lock.
D) The above code may lead to starvation.

22) Identify the correct statement from the following :
A) The Test-and-Set Lock hardware synchronization mechanism suffers from spin-lock when a round-robin scheduling algorithm is used.
B) Peterson's solution to the mutual-exclusion problem works for non-preemptive scheduling.
C) An operating system that can disable interrupts can implement semaphores.
D) None of these.

23) You are designing a data structure for efficient dictionary lookup in a multithreaded application. The design uses a hash table that consists of an array of pointers each corresponding to a hash bin. The array has 1001 elements, and a hash function takes an item to be searched and computes an entry between 0 and 1000. The pointer at the computed entry is either null, in which case the item is not found, or it points to a doubly linked list of items that you would search sequentially to see if any of them matches the item you are searching for. There are three functions defined on the hash table: Insertion (if an item is not there already), Lookup (to see if an item is there), and deletion (to remove an item from the table). Considering the need for synchronization, then choose the appropriate solution:

A) Use a mutex over the entire table
B) Use a mutex over each hash bin
C) Use a mutex over each hash bin and a mutex over each element in the doubly linked list
D) None of these

24) Which of the following are TRUE regarding the priority inversion problem? **[MSQ]**

A) Priority inversion may occur from resource synchronization among processes of different priorities.
B) Test-andSet lock suffers from priority inversion problems.
C) Priority inversion may lead to starvation.

D) Priority inversion problem may occur in any preemptive scheduling algorithms.

25) Consider the following code using mutex M1=1,M2=1, and M3=1.

| Process 1 | Process 2 | Process 3 |
|---|---|---|
| while (1) {<br>NonCriticalSection()<br>Mutex_lock(&M1);<br>Mutex_lock(&M2);<br>CriticalSection();<br>Mutex_unlock(&M2);<br>Mutex_unlock(&M1);<br>} | while (1) {<br>NonCriticalSection()<br>Mutex_lock(&M2);<br>Mutex_lock(&M3);<br>CriticalSection();<br>Mutex_unlock(&M3);<br>Mutex_unlock(&M2);<br>} | while (1) {<br>NonCriticalSection()<br>Mutex_lock(&M3);<br>Mutex_lock(&M1);<br>CriticalSection();<br>Mutex_unlock(&M1);<br>Mutex_unlock(&M3);<br>} |

Which of the following are correct ? **[MSQ]**

A) Mutual exclusion is satisfied.
B) Deadlock is possible only when all three resources (M1, M2 and M3) are held by processes.
C) Deadlock is not possible.
D) Deadlock is possible if only two resources out of M1, M2 and M3 are held by processes.

26) Which of the following are TRUE? **[MSQ]**

A) Lock variable synchronization mechanism ensures mutual exclusion.
B) Strict alternation synchronization mechanism ensures progress.
C) Peterson's Solution synchronization mechanism ensures bounded waiting.
D) Test-and-Set lock synchronization mechanism may suffer from deadlock.

27) Consider the following code, which is shared by four concurrent processes.

```
int count = 0, n=10; //Shared among all processes

int main()
{
  int i; // local to each process
  for(i=1;i<=n;i++){
    count=count+1;
```

```
  }
return 0;
}
```

The minimum value of count after completion of execution is
A) 1
B) 2
C) 10
D) 40

28) Consider the Compare-and-Swap synchronization mechanism to implement mutual exclusion on a critical section. Compare_and_Swap(&lock, oldval, newval) is atomic and writes newval into var and returns true if the old value of var is oldval. If the old value of var is not oldval, Compare-and-Swap returns false and does not change the value of the variable. Assume that the lock is initialized to 0.

```
int Compare-and-Swap(int *var,int oldval,int newval) {
    int temp = *var;
    if(*var == oldvalue)
       *var = newvalue
    return temp;
}
```

```
Entry{
  P:__________
}

//Critical Section

Exit{
 lock=0;
}
```

Choose the correct predicate P in the Entry section, to achieve mutual exclusion.
A) while( Compare-and-Swap(&lock, 1, 1)!=0);
B) while( Compare-and-Swap(&lock, 1, 0)!=0);
C) while( Compare-and-Swap(&lock, 0, 1)!=0);
D) None of these.

29) Which of the following is not a disadvantage of disabling interrupts to serialize access to a critical section
A) User code cannot utilize this technique for serializing access to critical sections
B) Interrupt controllers have a limited number of physical interrupt lines, thereby making it problematic to allocate them exclusively to critical sections.

C) This technique would lock out other hardware interrupts, potentially causing critical events to be missed.
D) This technique could not be used to enforce a critical section on a multiprocessor

30) What is the output produced by the following two processes P and Q using a muutex ?

| P | Q |
|---|---|
| While(1){<br>P(mutex);<br>print 1;<br>V(mutex);<br>} | While(1){<br>P(mutex);<br>print 2;<br>V(mutex);<br>} |

A) 1*2*
B) (12)*
C) (21)*
D) (1|2)*

31) Consider the following two concurrent processes

| int d = 0, r = 0; | |
|---|---|
| void p1 () {<br>d = 2;<br>r = 1;<br>} | int p2 () {<br>x=d;<br>while (!r) { }<br>return x;<br>} |

What are the possible return values of p2()?
A) 0, only.
B) 2, only
C) 0 or 2.
D) None of these.

32) Consider the following synchronization code using binary semaphore between two processes

| Int x=1, y=1<br>Semaphore mx = 1;my=0; | |
|---|---|
| P(mx); | P(my); |

| | |
|---|---|
| x = x+1<br>V(my); | x = y+1<br>V(mx); |

The value of x and y after execution is
A) infinite, 1
B) 2,1
C) 1,1
D) 3,1

33)

**1) Does starvation-freedom imply deadlock-freedom?**

Yes! If every process can eventually enter its critical section, although waiting time may vary, it means the decision time of selecting a process is finite. Otherwise, all processes would wait in the entry section.

**2) Does starvation-freedom imply bounded-waiting?**

No! This is because the waiting time may not be bounded even though each process can enter its critical section.

**3) Does bounded-waiting imply starvation-freedom?**

No! Bounded-Waiting does not say if a process can actually enter. It only says there is a bound. For example, all processes are locked up in the entry section (i.e., failure of Progress).

**4) We need Progress and Bounded-Waiting both to imply Deadlock-Freedom**

Yes! progress along with bounded waiting makes starvation-free, and starvation-free implies deadlock free.

**5) Progress does not imply Bounded Waiting:**

Yes! Progress says a process can enter with a finite decision time. It does not say which process can enter, and there is no guarantee for bounded waiting.

**6) Bounded Waiting does not imply Progress:**

Yes! Even though we have a bound, all processes may be locked up in the entry section (failure of Progress).