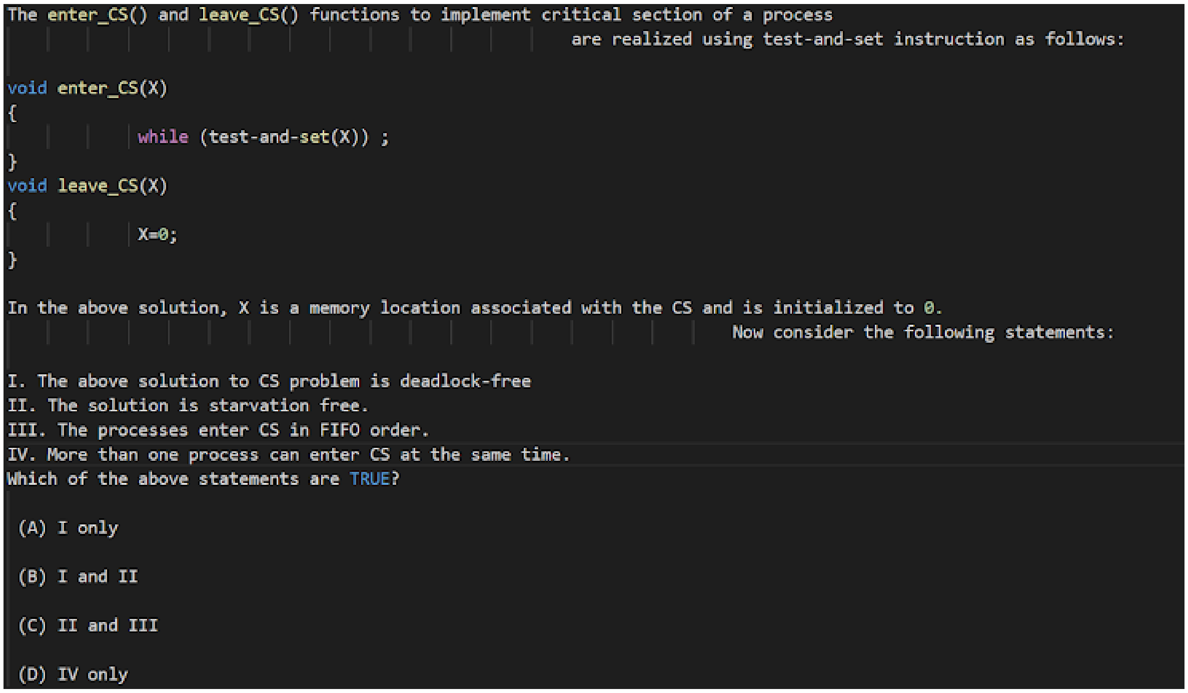
CS341-Operating System Quiz-3

Total points 38/100 ?

Email address *

maheeth2000@gmail.com

✕ 0/2



```
The enter_CS() and leave_CS() functions to implement critical section of a process
                                                        are realized using test-and-set instruction as follows:

void enter_CS(X)
{
            while (test-and-set(X)) ;
}
void leave_CS(X)
{
            X=0;
}

In the above solution, X is a memory location associated with the CS and is initialized to 0.
                                                                        Now consider the following statements:

I. The above solution to CS problem is deadlock-free
II. The solution is starvation free.
III. The processes enter CS in FIFO order.
IV. More than one process can enter CS at the same time.
Which of the above statements are TRUE?

 (A) I only

 (B) I and II

 (C) II and III

 (D) IV only
```

○ A

● B ✕

○ C

○ D

✓ 1/1

In resource allocation denial, a ……………………….. is one in which there is at least one sequence that does not result in a deadlock.
A) Safe state
B) Unsafe state
C) Safe allocation
D) Unsafe allocation

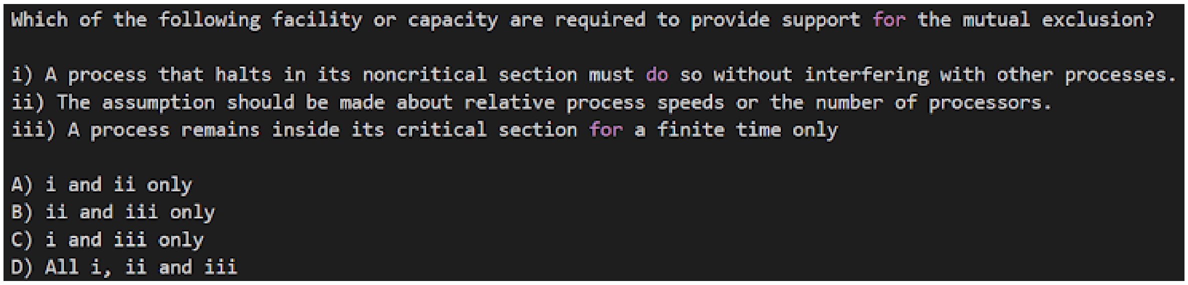
○ D

● A ✓

○ C

○ B

✗ 0/2



```
Which of the following facility or capacity are required to provide support for the mutual exclusion?

i) A process that halts in its noncritical section must do so without interfering with other processes.
ii) The assumption should be made about relative process speeds or the number of processors.
iii) A process remains inside its critical section for a finite time only

A) i and ii only
B) ii and iii only
C) i and iii only
D) All i, ii and iii
```

○ A

● B ✗

○ C

○ D

✓ Linux, uses spinlocks as a synchronization mechanism only on multiprocessor systems 1/1

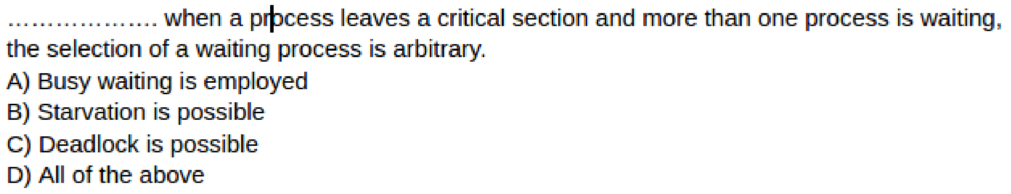
● True ✓

○ False

Roll Number *

1801CS31

✓ 1/1



……………… when a process leaves a critical section and more than one process is waiting, the selection of a waiting process is arbitrary.
A) Busy waiting is employed
B) Starvation is possible
C) Deadlock is possible
D) All of the above

○ A

● B ✓

○ C

○ D

✓ Data access synchronization ensures that shared data do not lose consistency when they are updated by interacting processes 1/1

● True ✓

○ False

✓ Control synchronization ensures that interacting processes perform their actions in a desired order 1/1

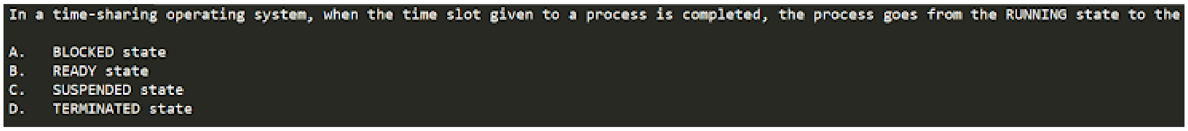
● True ✓

○ False

✗ ....is a queue of threads waiting for something inside a critical section …/2

Condition variable ✗

✓ 2/2



In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the RUNNING state to the

A. BLOCKED state
B. READY state
C. SUSPENDED state
D. TERMINATED state

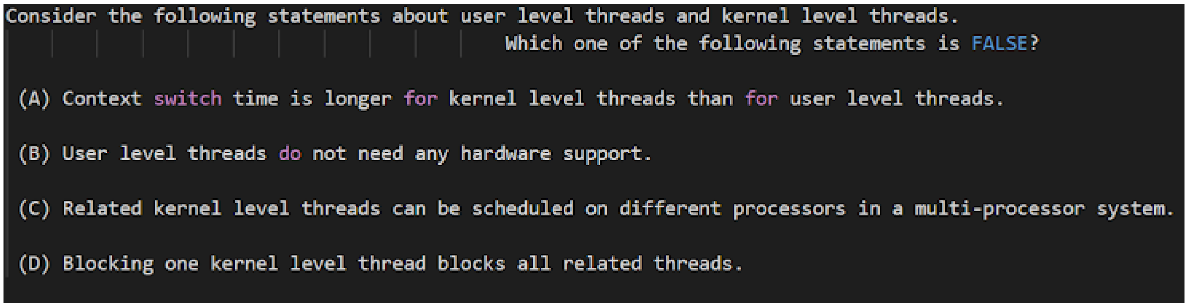
○ A

● B ✓

○ C

○ D

✓ 2/2



Consider the following statements about user level threads and kernel level threads.
Which one of the following statements is FALSE?

(A) Context switch time is longer for kernel level threads than for user level threads.

(B) User level threads do not need any hardware support.

(C) Related kernel level threads can be scheduled on different processors in a multi-processor system.

(D) Blocking one kernel level thread blocks all related threads.

○ A

○ B

○ C

● D ✓

✓ 1/1

Which of the following is known as uninterruptible unit
A) Single
B) Atomic
C) Semaphores
D) Static

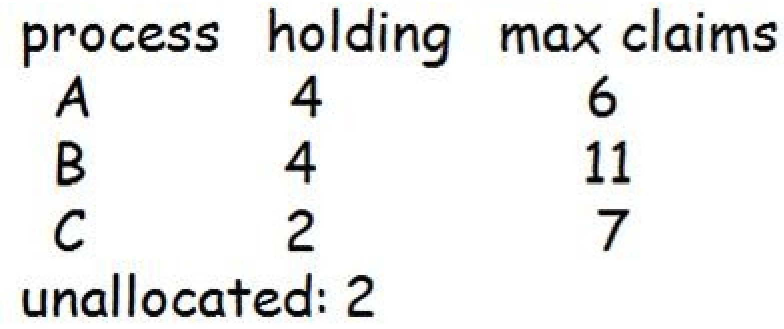
○ A

● B ✓

○ C

○ D

✗ Assuming one resource class. If C claims 9 instead of 7 . If safe sequence of execution is possible give sequence (Ans format : A, B, C) else give no safe sequence (Ans format : no/yes) …/3



process	holding	max claims
A	4	6
B	4	11
C	2	7

unallocated: 2

A, B, C ✗

✗ For implementing locks, ..., Disabling interrupts will only work on uniprocessors …/2

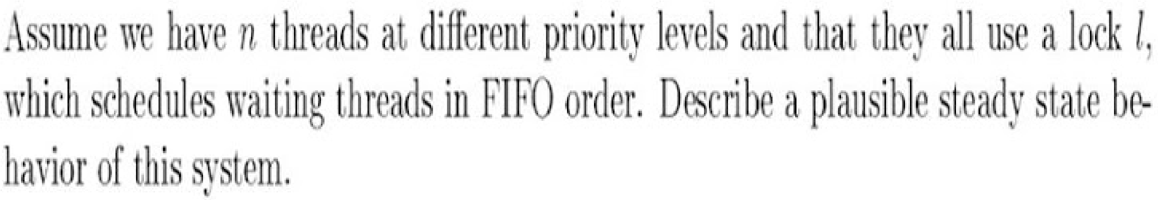
locks ✗

✓ Starvation implies deadlock 1/1

○ True

● False ✓

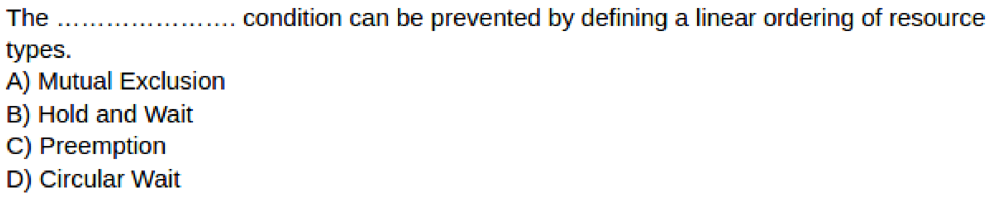
✗ Answer in short (possible example : We will lose all parallelism) …/5



Assume we have n threads at different priority levels and that they all use a lock l, which schedules waiting threads in FIFO order. Describe a plausible steady state behavior of this system.

✗

✗ 0/1



The ………………… condition can be prevented by defining a linear ordering of resource types.

A) Mutual Exclusion

B) Hold and Wait

C) Preemption

D) Circular Wait

○ A

● B ✗

○ C

○ D

✓ The algorithmic approach to implementing critical sections did not employ the process blocking and activation services of the kernel to delay a process 2/2

● True ✓

○ False

✗ 0/1

All processes share a semaphore variable **mutex**, initialized to 1. Each process must execute wait(mutex) before entering the critical section and signal(mutex) afterward.
Suppose a process executes in the following manner.

Signal (mutex)
....
Critical section
....
wait(mutex)

In this situation:
A) a deadlock will occur
B) processes will starve to enter critical section
C) several processes maybe executing in their critical section
D) all of the mentioned

○ A

○ B

○ C

● D ✗

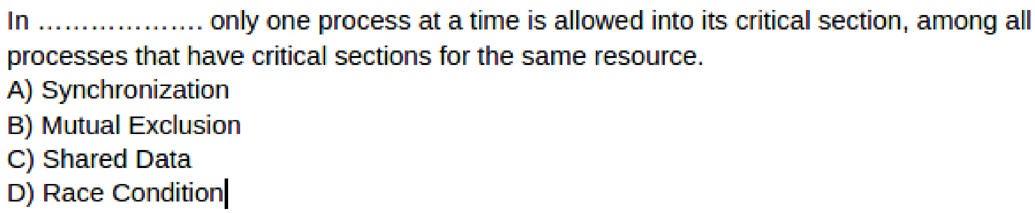
✗ Synchronization With.................., you don’t need to grab a mutex ···/2

concurrency ✗

✗ For implementing locks, ..., works on both uniprocessors and multiprocessors. ···/3

✗

✓ 1/1



In ………………. only one process at a time is allowed into its critical section, among all processes that have critical sections for the same resource.
A) Synchronization
B) Mutual Exclusion
C) Shared Data
D) Race Condition

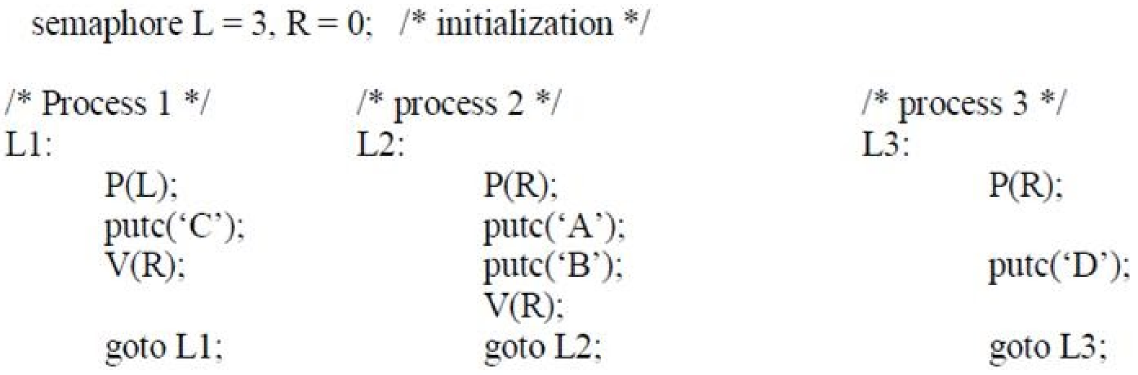
○ A

● B ✓

○ C

○ D

✕ In the following code, three processes produce output using the routine 'putc' and synchronise using semaphores "L" and "R". Is CABACDBCABDD a possible output sequence when this set of processes runs? …/4

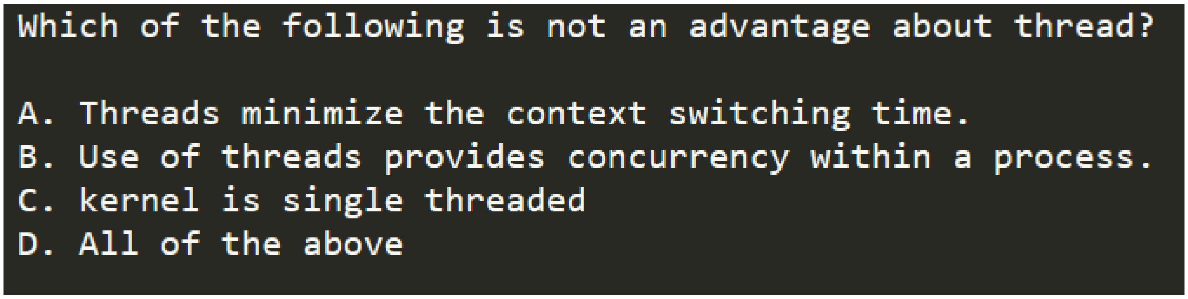


```
semaphore L = 3, R = 0;   /* initialization */

/* Process 1 */          /* process 2 */                /* process 3 */
L1:                      L2:                            L3:
        P(L);                    P(R);                          P(R);
        putc(‘C’);               putc(‘A’);
        V(R);                    putc(‘B’);                     putc(‘D’);
                                 V(R);
        goto L1;                 goto L2;                       goto L3;
```

✕

✗ 0/2



```
Which of the following is not an advantage about thread?

A. Threads minimize the context switching time.
B. Use of threads provides concurrency within a process.
C. kernel is single threaded
D. All of the above
```

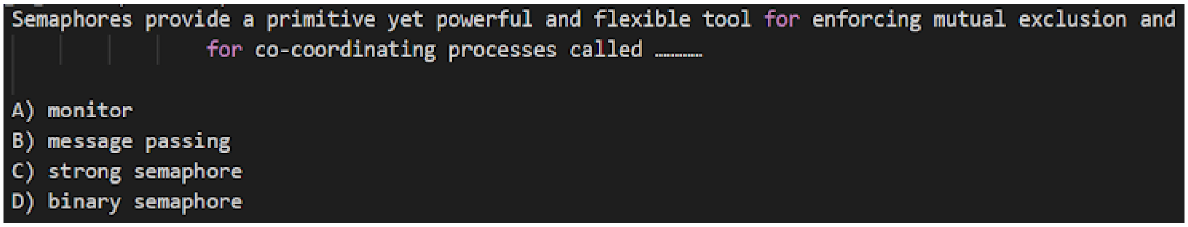
○ A

○ B

○ C

● D ✗

✓ 2/2



```
Semaphores provide a primitive yet powerful and flexible tool for enforcing mutual exclusion and
        for co-coordinating processes called ..........

A) monitor
B) message passing
C) strong semaphore
D) binary semaphore
```

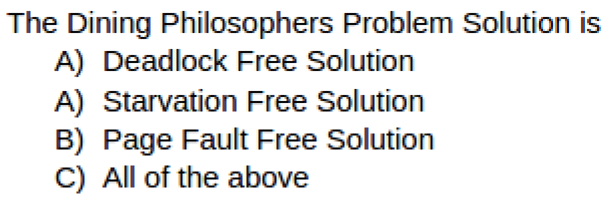
● A ✓

○ B

○ C

○ D

✗ 0/1



The Dining Philosophers Problem Solution is

A) Deadlock Free Solution

A) Starvation Free Solution

B) Page Fault Free Solution

C) All of the above

● A ✗

○ B

○ C

○ D

✗ .... is a strategy by which a user (or an application) exploits the characteristics of the CPU scheduling policy to get as much of the CPU time as possible. 0/2

● synchronization ✗

○ Round robin

○ time-sharing

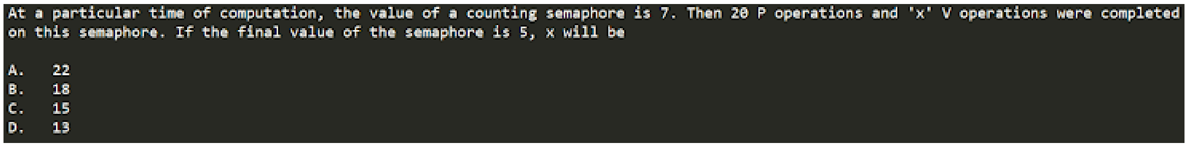
○ countermeasure

✓ The algorithmic approach to implementing critical sections did not employ indivisible instructions in a computer to avoid race conditions. 2/2

● True ✓

○ False

✓ 2/2



```
At a particular time of computation, the value of a counting semaphore is 7. Then 20 P operations and 'x' V operations were completed
on this semaphore. If the final value of the semaphore is 5, x will be

A.   22
B.   18
C.   15
D.   13
```

○ A

● B ✓

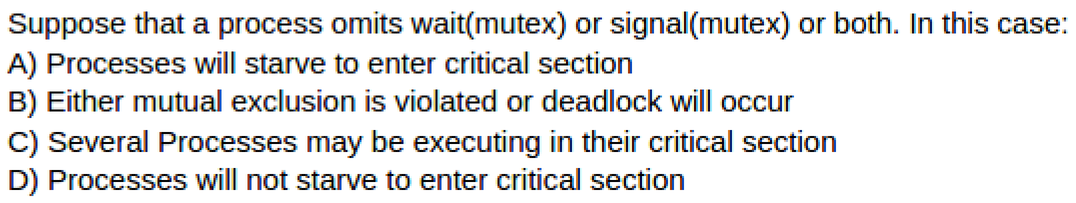
○ C

○ D

Name *

Maheeth Reddy

✓ 1/1



Suppose that a process omits wait(mutex) or signal(mutex) or both. In this case:
A) Processes will starve to enter critical section
B) Either mutual exclusion is violated or deadlock will occur
C) Several Processes may be executing in their critical section
D) Processes will not starve to enter critical section

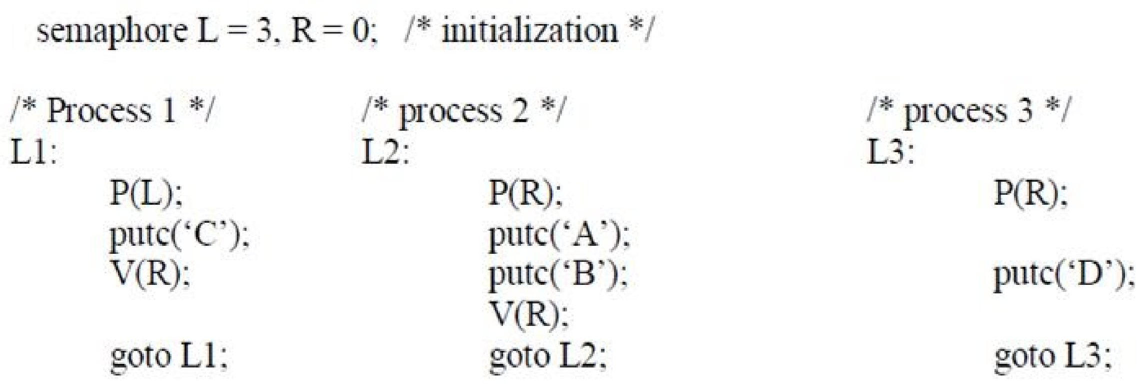
○ A

● B ✓

○ C

○ D

✗ In the following code, three processes produce output using the routine 'putc' and synchronise using semaphores "L" and "R". How any D's are printed when this set of process runs? (give answer in decimal number) …/4

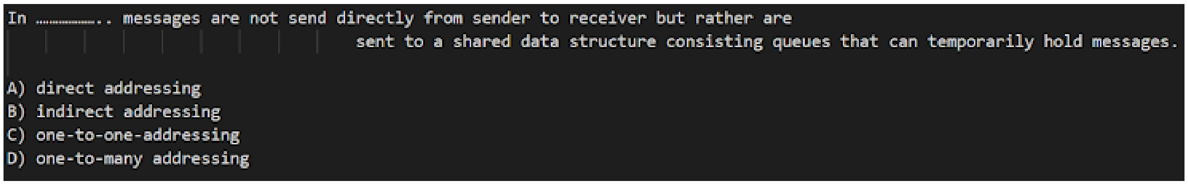


```
  semaphore L = 3, R = 0;   /* initialization */

/* Process 1 */          /* process 2 */              /* process 3 */
L1:                      L2:                          L3:
        P(L);                    P(R);                        P(R);
        putc('C');               putc('A');
        V(R);                    putc('B');                   putc('D');
                                 V(R);
        goto L1;                 goto L2;                     goto L3;
```

✗

✓ 2/2



```
In ………….. messages are not send directly from sender to receiver but rather are
                                   sent to a shared data structure consisting queues that can temporarily hold messages.

A) direct addressing
B) indirect addressing
C) one-to-one-addressing
D) one-to-many addressing
```

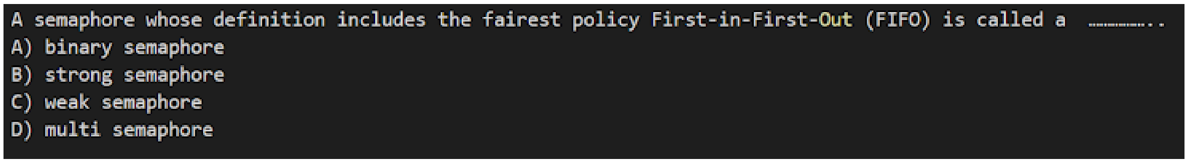
○ A

● B ✓

○ C

○ D

✓ 2/2



A semaphore whose definition includes the fairest policy First-in-First-Out (FIFO) is called a …………..
A) binary semaphore
B) strong semaphore
C) weak semaphore
D) multi semaphore

○ A

● B ✓

○ C

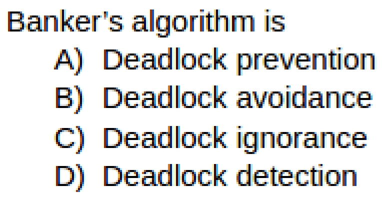
○ D

✓ Windows 2000 uses spinlocks as a synchronization mechanism in single processorsystems 1/1

○ True

● False ✓

✓ 1/1



Banker's algorithm is

A) Deadlock prevention
B) Deadlock avoidance
C) Deadlock ignorance
D) Deadlock detection

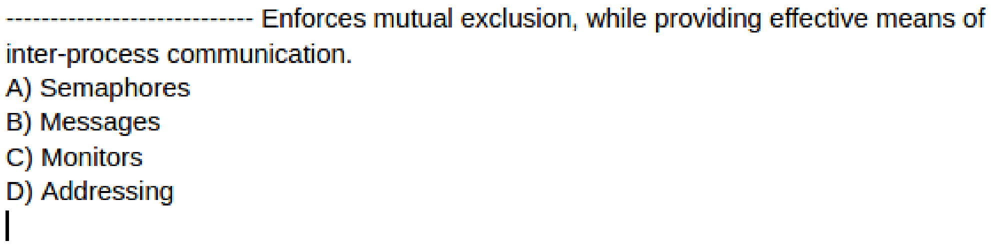
○ A

● B ✓

○ C

○ D

✓ 2/2



--------------------------- Enforces mutual exclusion, while providing effective means of inter-process communication.

A) Semaphores
B) Messages
C) Monitors
D) Addressing

○ A

● B ✓

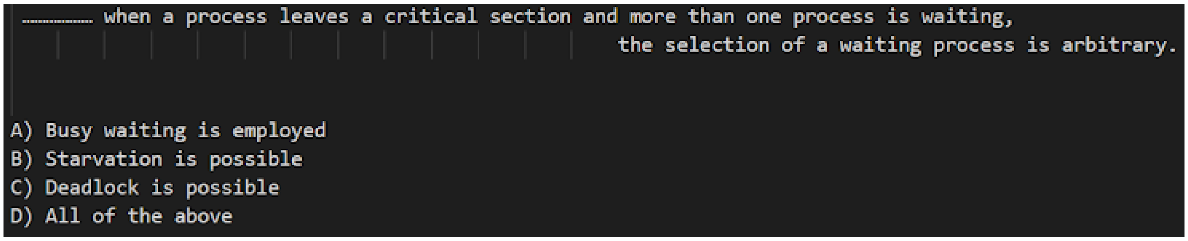
○ C

○ D

Find the best match

	approximate SRTF	busy waiting	Java	synchronous events in the machine	tasks are placed into the lowest-priority queue	Threads	Score	
Monitor	☑	□	□	□	□	□	0/1	›
Traps	□	☑	□	□	□	□	0/1	›
spinlocks	□	□	☑	□	□	□	0/1	›
MLFQ	□	□	□	☑	□	□	0/1	›

✓ 2/2



```
 ............... when a process leaves a critical section and more than one process is waiting,
                                          the selection of a waiting process is arbitrary.


A) Busy waiting is employed
B) Starvation is possible
C) Deadlock is possible
D) All of the above
```

○ A

● B ✓

○ C

○ D

✗ In .... jobs are always put on the highest priority queue when they become ready to run ···/2

priority queue ✗

✓ We use dynamic memory because storing data on the stack requires knowing the size of that data at compile time 2/2

● True ✓

○ False

Find the best match

	stop one of four necessary conditions for deadlock	Assesses, for each allocation, whether it has the potential to lead to deadlock	Attempts to assess whether waiting graph can ever make progress	Ignore the problem and pretend that deadlocks never occur in the system	Allow system not to enter deadlock	Score	
Techniques for addressing Deadlock	☑	□	□	□	□	0/1	✗
Deadlock prevention	□	☑	□	□	□	0/1	✗
Deadlock avoidance	□	□	☑	□	□	0/1	✗
Deadlock detection (next time) and recover	□	□	□	☑	□	0/1	✗

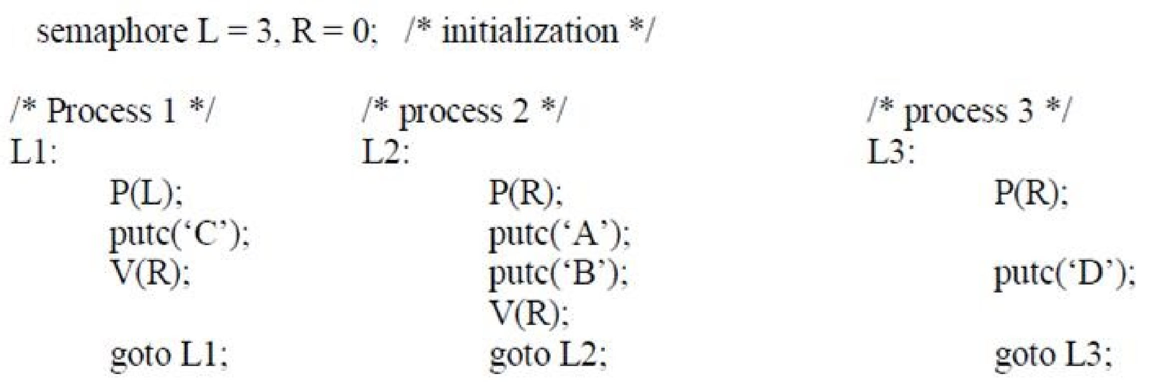
Match the following

	Piece of code that only one thread can execute at once	Ensuring that only one thread does a particular thing at a time	Isolating program faults to an address space	Using atomic operations to ensure cooperation between threads	Score	
Synchronization	☑	□	□	□	0/1	✗
Mutual exclusion	□	☑	□	□	1/1	✓
Critical section	□	□	☑	□	0/1	✗

Find the best match

	not require programmers to recheck conditions	'if' statement	“while” loop	signaling thread simply placed the signaled thread on the run queue and continues executing	signal() operation from one thread immediately wakes up a sleeping thread, hands the lock to the sleeping thread, and starts the sleeping thread executing;	Score	
Hoare scheduling	☑	☐	☐	☐	☐	0/1	✗
Mesa-scheduling	☐	☑	☐	☐	☐	0/1	✗
Mesa-monitors	☐	☐	☑	☐	☐	1/1	✓
Hoare-monitors	☐	☐	☐	☑	☐	0/1	✗

✕ In the following code, three processes produce output using the routine 'putc' and synchronise using semaphores "L" and "R". Is CABABDDCABCABD a possible output sequence when this set of processes runs? /4

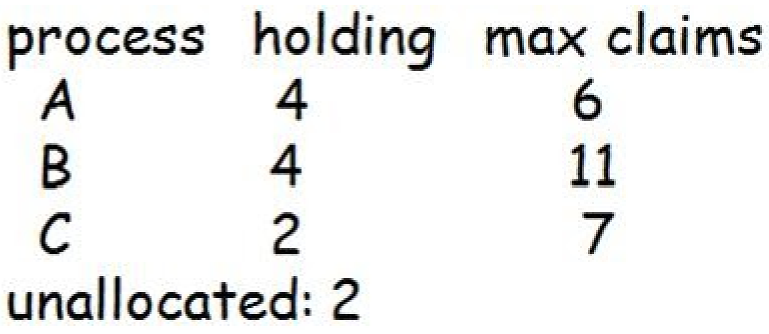


```
   semaphore L = 3, R = 0;   /* initialization */

/* Process 1 */          /* process 2 */                    /* process 3 */
L1:                      L2:                                L3:
        P(L);                     P(R);                              P(R);
        putc('C');                putc('A');
        V(R);                     putc('B');                         putc('D');
                                  V(R);
        goto L1;                  goto L2;                           goto L3;
```

✕

✕ Assuming one resource class. If safe sequence of execution is possible give sequence (Ans format : A, B, C) else give no safe sequence (Ans format :yes/no /3



process	holding	max claims
A	4	6
B	4	11
C	2	7

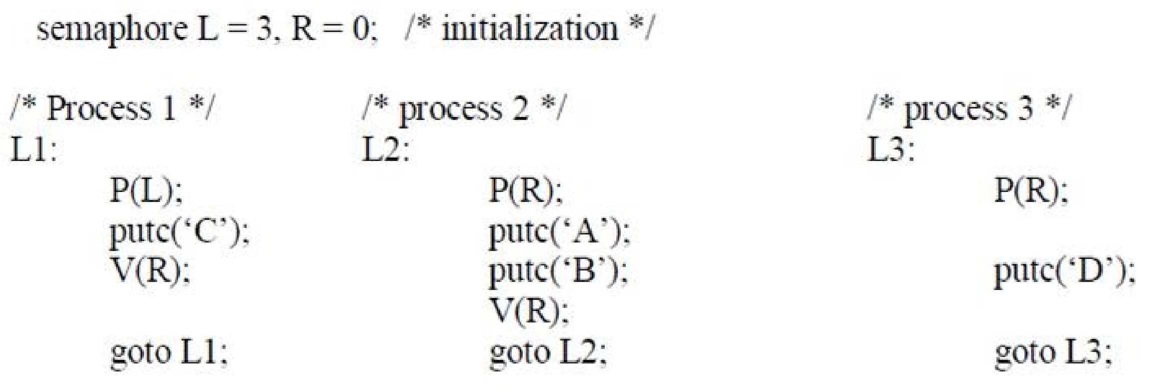
unallocated: 2

A, B, C ✕

Find the best match

	It is much easier to share resources, such as memory or file handles	It is not easier to share resources, such as memory or file handles	The processor may not know how to schedule them, if they aren't kernel-threads	A corruption by one thread can not impact other threads	Allow system not to enter deadlock	Score	
Threads	☑	□	□	□	□	1/1	✓
process	□	☑	□	□	□	1/1	✓
Threads have the disadvantage that	□	□	☑	□	□	1/1	✓

✗ In the following code, three processes produce output using the routine 'putc' and synchronise using semaphores "L" and "R". What is the smallest number of A's that might be printed when this set of process runs (give answer in decimal number) …/4



```
  semaphore L = 3, R = 0;   /* initialization */

/* Process 1 */            /* process 2 */                /* process 3 */
L1:                        L2:                            L3:
        P(L);                      P(R);                          P(R);
        putc('C');                 putc('A');
        V(R);                      putc('B');                     putc('D');
                                   V(R);
        goto L1;                   goto L2;                       goto L3;
```

5 ✗