|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Apellido y Nombre | Eje. | 1) 6 | 2) 5 | 3) 5 | 4) 6 | 5) 6 | 6) 6 | 7) 6 | 8) 20 | 9) 20 | 10) 20 | Nota: |
|  | Pun. |  |  |  |  |  |  |  |  |  |  |  |

Recuperatorio Segundo Parcial

# Indique Verdadero o Falso:

1. Si se aplica un algoritmo que va tomando procesos teniendo en cuenta el menor tiempo de ejecución puede darse Livelock
2. Un semáforo evita que un proceso realice una espera activa en la memoria de disco mientras llega su turno de ejecución.
3. El interbloqueo se produce cuando varios procesos quieren acceder al mismo recurso y el semáforo no se los permite.

# Indique, con un círculo o una cruz, cuál/es es/son la/s opción/es correcta/s:

1. En la solución al problema clásico del Productor-Consumidor:
	1. Hay un consumidor que está tomando ítems del buffer
	2. Hay uno o más productores generando algún tipo de dato y colocándolo en un buffer como así también quitándolos.
	3. Se debe asegurar que el productor pueda agregar datos si el buffer está lleno y que el consumidor intente remover datos, aunque esté vacío.
2. Un proceso que ejecuta un SemWait con valor 2:
	1. Se envía a LISTOS.
	2. Se envía a BLOQUEADOS.
	3. Queda en CPU.
	4. Ninguna de las anteriores.
3. Señale uno de los requisitos que considere correcto para satisfacer el buen funcionamiento del algoritmo que resuelve el problema de la sección crítica:
	1. Un proceso debe bloquearse mientras avanza dentro de la región crítica, justo antes de terminar
	2. Un proceso debe sufrir inanición esperando acceder a la región crítica
	3. Cada proceso puede ejecutar el código de esa sección simultáneamente siempre y cuando no se pisen sobre la misma línea.
	4. Ninguna de las anteriores
4. Dado el valor del semáforo con contador = -1 puedo afirmar que:
	1. No hay ningún proceso bloqueado.
	2. Que hay un proceso bloqueado.
	3. Que hay al menos un proceso bloqueado.

## Se tienen los siguientes tres procesos: curly, larry y moe, que son ejecutados concurrentemente. Inicializar los semáforos que considere necesarios y sincronizarlos de tal forma que X (variable global) siempre muestre:

100

|  |  |  |
| --- | --- | --- |
| void curly() { while (true) {X= 200; X= X/2;Printf(X);}} | void larry() { while (true) { X= 400;X= 400/2;Printf(X);}} | void moe() { while (true) {X= 150; X= 150\*2;Printf(X);}} |

## 200

300

## 100

200

## 300

9- Analice el siguiente fragmento de pseudo-código escrito en C y responda

a) ¿Terminan todos los hilos? ¿Por qué?

b) ¿Cuál es el resultado que se muestra por pantalla? ¿Cómo se llegó a ese resultado?

c) ¿Qué pasa si no estuviese la función pthread\_join?

d) Explique brevemente: pthread\_join, pthread\_t, pthread\_exit, pthread\_self

Variable global: int a = 8;

Funciones:

\*f1(void \*arg) {

1. = 2;

println("Thread: %d - valor de a: %d", pthread\_self(), a); }

\*f2(void \*arg) { pthread\_exit(0);

a\* = 2;

println("Thread: %d - valor de a: %d", pthread\_self(), a); }

void main() {

pthread\_t h1, h2; println("Thread: %d", pthread\_self());

pthread\_create(&h2, NULL, f2, NULL);

pthread\_create(&h1, NULL, f1, NULL);

pthread\_join(h1, NULL);

pthread\_join(h2, NULL);

return(0);