CD Examen Teoría y Práctica **19 de diciembre de 2005**

Dr. Arno Formella, Dra. María José Lado Touriño

Apellidos, Nombre: _____

D.N.I: _____

Firma: _____

Prácticas presentadas: _____

_____

1	2	3	4	5	6	7	8	9	Suma
(4)	(6)	(8)	(8)	(4)	(6)	(2)	(4)	(8)	(50)

Pregunta 1: [4 Puntos]

Razona como están relacionadas la distribución del trabajo con la distribución de los datos en una aplicacion concurrente en un sistema distribuido.

Pregunta 2: [6 Puntos]

Describe brevemente la posibilidad usando el "test-and-set" como ayuda en hardware para implementar acceso con exclusión mutua a la memoria y destaca su diferencia con el algoritmo de Dekker que solamente trabaja con `load` y `store`. (Describe también dicho algoritmo de Dekker.)

Pregunta 3: [8 Puntos]

¿Cuáles son las condiciones que se tienen que cumplir para que se produzca un bloqueo entre procesos? Describe los tres métodos disponibles para solventar el problema del bloqueo. Razona brevemente sobre su eficiencia en entornos distribuidos.

Pregunta 4: [8 Puntos]

En una aplicación se tiene que gestionar tres tipos de procesos/hilos con diferentes prioridades (digamos A, B y C) que quieren acceder a un único recurso. ¿Cómo implementarías el control del planificador para que todos los procesos tengan acceso al recurso con la siguiente forma de justicia: dentro de la misma prioridad el acceso se realiza en orden de pedido y entre los diferentes prioridades se distribuye los accesos para que a lo largo del tiempo por lo menos unos 50 % de los accesos son para los procesos de tipo A, 35 % para los del tipo B y 15 % para los del tipo C? Razona si tu solución garantiza una espera *finita* para todos los procesos pidiendo acceso al recurso.

Pregunta 5: [4 Puntos]

Describe brevemente el patrón de diseño *reactor*.

Pregunta 6: [6 Puntos]

Describe un protocolo de entrada y salida para varios hilos que garantiza exlusión mutua sin espera activa.

Pregunta 7: [2 Puntos]

¿Cómo se puede conseguir que el código de un constructor se ejecute con exclusión mutua en Java?

Pregunta 8: [4 Puntos]

Explica la semántica del modificador `volatile` de Java y su uso en programas concurrentes. ¿Cómo evita dicha semántica introducida en la versión 1.5 de Java que el optimizador haga reordenaciones del código inesperadas por el usuario?

Pregunta 9: [8 Puntos]

Describe brevemente pero de forma precisa como implementamos las operaciones *insertar*, *borrar*, e *iterar* sobre una lista concurrente que tenga la propiedad que varios hilos pueden manipular la lista concurrentemente en zonas suficientemente separadas.