CD Examen Teoría/Prácticas **Dr. Arno Formella, Julio de 2014**

Apellidos: _____ **Nombre:** _____

D.N.I: _____ **Firma:** _____

1	2	3	4	5	6	7	8	9	10	Suma
(4)	(6)	(4)	(4)	(6)	(6)	(10)	(4)	(3)	(3)	(50)

Pregunta 1: [4 Puntos] ¿Por qué puede ser ventajoso desarrollar un programa concurrente incluso en un entorno de desarrollo mono-procesador? (Piensa como un ingeniero informática en fase de análisis de diseño de software...)

Pregunta 2: [6 Puntos] Describe brevemente el funcionamiento de la instrucción hardware TAS disponible para muchos procesadores. ¿Para qué se puede utilizar? ¿Se puede implementar un protocolo de entrada/salida a una sección crítica con TAS que permite dar paso de un proceso a otro (en caso que sí razona cómo, en caso que no razona por qué no)?

Pregunta 3: [4 Puntos] Describe brevemente el patrón de diseño *aviso de hecho* (*double scoped locking*). Razona críticamente sobre su implementación en Java, C++03 y C++11.

Pregunta 4: [4 Puntos] ¿Cuáles son las condiciones que se tienen que cumplir para que se produzca un bloqueo entre procesos?

Pregunta 5: [6 Puntos] Alguien te ofrece unos protocolos de entrada y salida para garantizar la exclusión mutua de una sección crítica involucrando dos procesos. ¿Cómo emplearías el principio de la bandera para comprobar ciertos (¿cuáles?) aspectos del protocolo?

Pregunta 6: [6 Puntos] Explica la semántica del modificador `volatile` de Java y su uso en programas concurrentes. Entre otras, qué tiene que ver con una relación *ha-pasado-antes*, es decir, qué garantías tiene el programador con el uso correcto de `volatile`,

Pregunta 7: [10 Puntos] Reflexiona brevemente sobre las posibilidades y limitaciones para la programación concurrente en el uso de `synchronized` en Java. Sugiere algunas mejoras (directamente como amplicación de la notación de Java) para superar aspectos negativos de la implementación actual en Java.

Pregunta para las prácticas:

Pregunta 8: [4 Puntos] En ciertas ocaciones medimos el tiempo de ejecución de un programa concurrente y queriamos documentar la influencia del número de hilos participantes y del tamaño del problema (p.ej. número de operaciones). ¿Cuáles fueron las principales técnicas empleadas para medir? ¿Cuáles fueron las principales dificultades con las cuales uno se tiene que enfrentar cuando se interpreta los datos de la medición? ¿Cuáles fueron tus observaciones en el caso de *salida a pantalla* y *operaciones internas* de la CPU en un ordenador con multi-nucleo?

Pregunta 9: [3 Puntos] En el ejercicio del PingPong vimos que el uso de `notify()` y `notifyAll()` resultaba en un número de jugadas entre los jugadores en cierto intervalo de tiempo que dependía del número de jugadores participando en el juego. ¿Por qué eso era el caso? ¿Cómo conseguimos finalmente un número constante de jugadas, en cierto intervalo de tiempo, independientemente del número de jugadores participantes? ¿Cómo implementarías un método para medir bien el número de jugadas en cierto tiempo (sin medir, dentro de lo posible, la influencia de otras partes del programa)?

Pregunta 10: [3 Puntos] En las prácticas hemos visto dos tipos de estructuras de datos concurrentes: una implementación propia de una lista concurrente que dispone de las operaciones *insertar*, *borrar*, e *iterar*, y una tabla de dispersión que está disponible en el paquete `java.util.concurrent` llamada `ConcurrentHashMap`. Describe brevemente cuáles son las principales características de las dos implementaciones.