CDI Examen Teoría **Dr. Arno Formella, mayo de 2017**

Apellidos: _____ **Nombre:** _____

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

1	2	3	4	5	6	7	8	9	10	11	12	13	14	Suma
(5)	(5)	(5)	(5)	(5)	(5)	(4)	(4)	(4)	(4)	(4)	(4)	(4)	(2)	(30/60)

Lo/as estudiantes **con** evaluación continua deben contestar solamente las primeras **seis** (6) preguntas (Parte I). Tiempo: 1 hora y media.

Lo/as estudiantes **sin** evaluación continua deben contestar todas las **doce** (14) preguntas (Parte I+II+III). Tiempo: 3 horas y media.

Una vez empezado el examen, el/la estudiante acepta que cualquier uso de un dispositivo móvil sin previo aviso al profesor, resulta en un suspenso inmediato del examen con puntuación 0, informe a los órganos competentes de la Universidad de un posible intento de fraude, y posibilidad a un futuro examen en este curso solamente mediante orden escrita del órgano superior.

Parte I

Pregunta 1: [5 Puntos] La *región crítica* es un concepto abstracto útil para la programación concurrente. Este concepto está disponible en el lenguaje de programación Java. Detalla su síntaxis y uso. Razona críticamente sobre posibles ventajas y desventajas, tanto a nivel del propio concepto como herramienta de programación concurrente, como a nivel de implementación en el lenguaje Java. ¿Es fácil llevar el concepto a un entorno distribuido?

Pregunta 2: [5 Puntos] En clase hemos visto un protocolo de entrada y salida a una sección crítica para dos procesos donde los procesos ejecutan código diferente (protocolo asimétrico).

1. Detalle el protocolo en pseudo código.
2. Comprueba la corrección de la exclusión mutua.
3. Razona sobre la política de justicia de este protocolo.

Pregunta 3: [5 Puntos] Detalle tres implementaciones diferentes como realizar (en Java) un contador concurrente (valores enteras de 64 bit) donde varios hilos pueden incrementar el valor del contador concurrentemente. Argumenta por qué con una declaración simple del entero como *volatile* no funciona correctamente.

Pregunta 4: [5 Puntos] Implementa en pseudo código con la instrucción hardware CAS un protocolo de entrada y salida a una sección crítica para dos procesos con garantía de espera finita para los dos procesos participantes. Ayuda: recuerda el protocolo de Dekker (en su quinto intento).

Pregunta 5: [5 Puntos] ¿En la cuarta actividad (lectura de un libro con varios hilos), existe en la solución donde el último hilo que ha leído una página despierta el siguiente hilo por leer la siguiente página la posibilidad de un bloqueo entre hilos? En caso que no, ¿cuál es el argumento? (No te olvides hablar del caso de un sólo lector que lea todo el libro!)

Pregunta 6: [5 Puntos] Durante la fase de depuración de un programa concurrente, a la responsable de realizar las pruebas ocurre la idea de producir una versión del código introduciendo simples comandos de `sleep` de cierta duración justamente delante de todos los `wait` ya existentes en el programa. ¿Debería funcionar el programa lógicamente igual? En caso que sí, ¿qué tipo de fallos se pueden detectar, si el programa se comporta diferente? ¿Te ocurre algún inconveniente de tal prueba (a parte que las pruebas siempre consumen tiempo en la producción de una aplicación software)?

Parte II

Pregunta 7: [4 Puntos] Explica que es una condición de carrera. Añade un posible ejemplo y una solución para tu ejemplo.

Pregunta 8: [4 Puntos] Explica que es una espera activa y sus inconvenientes. Añade un posible ejemplo y razona como se puede evitar. ¿Se puede evitar en todos los casos?

Pregunta 9: [4 Puntos] ¿Cuáles son las condiciones que se tienen que cumplir para que se produzca un bloqueo entre procesos? ¿Cómo se puede prevenir la aparencia de un bloqueo entre procesos en muchas ocaciones ya durante el diseño de una aplicación? ¿Te puedes imaginar una situacion donde no se puede prevenir? Detalla tal situación y propon una posible solución.

Pregunta 10: [4 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/a programador/a con el uso de `volatile` escribiendo y leyendo variables en su código?

Pregunta 11: [4 Puntos] Describe brevemente cuatro (4) características disponibles en los paquetes `java.util.conconcurrent` y `java.util.conconcurrent.lock`. Destaca en cada caso su semántica princial y su especial relevancia para su uso en programas concurrentes.

Parte III

Pregunta 12: [4 Puntos] Describe como has implementado la medición de tiempo de ejecución de tu programa concurrente. ¿Qué fases del programa hay que distinguir cuya combinación resulta en el tiempo global? ¿Cúales fueron tus tiempos de ejecución medidos (a grandes rascos) para el problema con la operación sobre imágenes grandes y para el problema de la lectura concurrente del libro aumentando gradualmente el número de hilos participantes en ambos casos?

Pregunta 13: [4 Puntos] ¿Para que sirve un `CountDownLatch`? Explica su posible uso en alguna de las actividades prácticas propuestas durante el curso.

Pregunta 14: [2 Puntos] El manual de Java comenta que existe la posibilidad de que un hilo se despierta de un `wait` por razones desconocidas (*spurious wakeup*), es decir, el hilo termina el `wait` sin haber recibido un `notify`. ¿Qué solución propones para este posible problema que puede ocurrir en una máquina virtual de Java.