PCTR - PREGUNTAS TIPO TEST EXAMENES (2017-2018)

1- **[FEB 2018]** El multiprocesamiento simétrico, contemplado en la jerarquía de Flynn en el nivel MIMD
   a) Engloba a arquitecturas multicore
   b) Engloba a arquitecturas multicore y manycore
   c) Engloba a arquitecturas multicore, manycore, a excepción los clusters de procesadores
   d) En realidad todo lo anterior es cierto. El multiprocesamiento simétrico es tan general como para englobar a todas las tipologías de máquinas que disponen de múltiples cores de ejecución, con independencia de su número y organización.

*El multiprocesamiento simétrico es sinónimo de procesamiento concurrente de grano grueso, cada hebra se reparten el mismo número de tareas

1- **[FEB 2017]** Los hilos creados por otro hilo
   a) No necesitan implementar acceso a exclusión mutua para compartir recursos
   b) No necesitan exclusión mutua siempre y cuando el padre acceda a dichos recursos en exclusión mutua.
   c) Siempre deben acceder a los recursos compartidos en exclusión mutua
   d) Ninguna es correcta

2- **[FEB 2017]** Los hilos creados por otro hilo pueden compartir información con el hilo padre
   a) A través de variables estáticas definidas en el hilo padre o el hilo hijo
   b) Pasando por referencia a los hilos hijos las variables compartidas
   c) A través de un archivo
   d) Todas son correctas

3- **[FEB 2017]** Cada hilo comparte con su hilo padre
   a) Contador de programa
   b) Contador de programa y la pila de llamadas
   c) La pila de llamadas y las variables estáticas
   d) Ninguna es correcta

4- **[FEB 2017]** Un hilo creado por otro hilo
   a) Es siempre reentrante, por lo que no se bloquea al realizar llamadas recursivas
   b) Nunca se bloquea al tratar de adquirir un cerrojo bloqueado por el hilo padre, pues comparten espacio de direcciones.
   c) Debe esperar para poder adquirir cerrojo de los métodos sincronizados del hilo padre
   d) Ninguna es correcta

5- **[FEB 2017]** El entrelazado de instrucciones
   a) Solo se produce entre hilos, pero no entre procesos
   b) Solo se produce en arquitecturas multinucleo
   c) Se produce en sistemas concurrentes y paralelos
   d) No se producen cuando se ejecutan instrucciones del sistema operativo

6- **[FEB 2017]** ¿Cuál es el número máximo de clases que pueden heredar de la clase Thread en una aplicación en Java?
   a) No hay limite
   b) Solo una clase puede heredar de Thread. El resto de ben implementar la interfaz Runnable
   c) Solo una, si se desea compartir información entre los hilos a través de variables estáticas
   d) Todas las clases deben heredar de la clase Thread

7- **[FEB 2017]** El acceso en exclusión mutua a los recursos compartidos
   a) Produce aplicaciones menos eficientes
   b) Produce aplicaciones más eficientes
   c) No afecta al rendimiento de las aplicaciones
   d) Produce aplicaciones más eficientes cuando se utilizan primitivas de sincronización reentrantes

8- **[FEB 2017]** Si se aplica en Java la instrucción H.notify()
   a) La señal se perderá si el hilo H no se encuentra bloqueado.
   b) La señal se perderá si no hay ningún hilo en el Wait Set asociado a la variable H.
   c) Se incrementa en una unidad el número de veces que se puede realizar H.wait() sin que el hilo se quede bloqueado.
   d) Ninguna es correcta.

9- **[FEB 2017]** Dadas dos variables A y B en Java, que apuntan a la misma dirección de memoria (A=B)
   a) Cada una de ellas dispondrá de su propio Wait Set asociado
   b) Cada una de ellas dispondrá su propio Wait Set asociado si ambas son de tipo Object
   c) Ambas variables comparten el mismo Wait Set asociado
   d) Ambas variables comparten el mismo Wait Set asociado pero solo si ambas son de tipo Object

11- [FEB 2017] Los métodos bloqueantes que se pueden aplicar sobre un hilo en Java son:

   a) join,suspend
   b) join, wait, yield
   c) Sleep, join, wait
   d) Sleep, join, wait, suspend

Wait, join y sleep son bloqueantes

12- **[FEB 2017]** Los hilos que se ejecutan en un pool de hilos no requieren de un acceso en exclusión mutua a los recursos compartidos

a) Cuando se utiliza un pool de tipo CachedThreadPool
b) Cuando se utiliza un pool de tipo FixedThreadPool
c) Cuando se utiliza un pool de tipo SingleThreadExecutor
d) Ninguna es correcta

13- **[FEB 2017]** Un hilo…

a) Puede estar ejecutando el protocolo de entrada a la sección crítica mientras otro se encuentra ejecutando instrucciones del protocolo de salida.
b) No puede estar ejecutando el protocolo de entrada a la sección crítica mientras otro se encuentra ejecutando instrucciones del protocolo de salida.
c) Puede encontrarse ejecutando instrucciones de la sección crítica mientras otro hilo ejecuta las del protocolo de salida.
d) Puede estar ejecutando instrucciones del protocolo de salida al mismo tiempo que otro hilo también las ejecuta.

14- **[FEB 2017]** Los protocolos de exclusión mutua basados en soluciones software

a) Solo funcionan correctamente si se accede en exclusión mutua a las variables compartidas
b) Solo funcionan correctamente si se accede por turnos a las variables compartidas
c) Son más eficientes que los basados en soluciones hardware
d) Ninguna es correcta

15- **[FEB 2017]** El método shutDownNow() de un objeto de la clase ExecutorService

a) Es un método equivalente a invocar al método shutdown() seguido de una llamada al método awaitTermination()
b) Es un método bloqueante equivalente a invocar el método shutdown() seguido de un blucle que finalice cuando el método isTerminated() devuelva un valor cierto
c) Es un método bloqueante que espera a la finalización de los hilos que actualmente se encuentran ejecutándose en el pool de hilos, pero no al resto de hilos que se encuentran pendientes de ejecución.
d) Es un método que finaliza los hilos del pool de forma inmediata, sin esperar a la finalización de los que actualmente se ejecutan.

Diferencias entre shutdown() -> Indica al ejecutor que no se van a mandar más tareas y que espere a que finalicen el resto y shutdownNow() finaliza inmediatamente todos los hilos

16- **[FEB 2017]** Cuando se utilizan métodos sincronizados para implementar un monitor en Java, todos sus métodos

a) Deben ser sincronizados
b) Deben ser públicos y sincronizados
c) Deben ser públicos, estáticos y sincronizados
d) …que sean públicos deben ser sincronizados

17- **[FEB 2017]** Los monitores en C++

a) Solo pueden implementarse mediante el uso de instancias de tipo recursive_mutex
b) Pueden implementarse mediante el uso de instancias de tipo unique_lock
c) Solo pueden implementarse mediante el uso de primitivas reentrantes
d) Pueden implementarse mediante el uso de la primitiva de sincronización call_once

18- [FEB 2017] Un método sincronizado en Java

a) Puede contener un bloque sincronizado que utilice this como cerrojo
b) Puede contener un bloque sincronizado que utilice una variable estática como cerrojo
c) No puede contener un bloque sincronizado
d) Las respuestas a y b son correctas

19- [FEB 2017] En el modelo de memoria CUDA

a) Las regiones de memoria per-block requieren control de exclusión mutua
b) Existen regiones de memoria que no requieren control de exclusión mutua
c) La región de memoria global requiere control de exclusión mutua
d) Todas las respuestas son correctas

Memoria local -> no em, memoria compartida-> em, memoria global -> em

20- [FEB 2017] En la especificación JRTS la interfaz Schedulable

a) Modela las clases RealTImeThread y NoHeapRealTimeThread
b) Hereda de la interfaz Runnable de Java
c) Modela la clase AsyncEventHandler
d) Todas las respuestas anteriores son correctas

21- [FEB 2017] En el framework RMI el servidor

a) Puede efectuar la llamada a un método del cliente, siempre y cuando el cliente quede asociado a un nombre de servicio en el registro de RMI local
b) Puede efectuar dicha llamada, solo si el cliente implementa la interfaz Serializable
c) Solo puede disponer de métodos públicos
d) Ninguna es correcta

22- [FEB 2018] Suponga que 1000 hebras comparten el acceso compartido a un objeto común cada una a través de su propia referencia, y ejecutan el acceso a un método inc() de ese objeto que incrementa un contador que inicialmente vale -1. El programador protege el acceso haciendo que cada hebra ejecute la llamada a inc() envolviendo la misma en un bloque de la forma synchronized(this){…miReferencia.inc()…}. El valor final del contador es:

a) 999
b) 1000
c) Un número indeterminado entre 0 y 1000
d) Un número indeterminado entre 1 y 1001
e) Ninguna de las anteriores es correcta

23- [FEB 2018] Considere un objeto en Jaba que dispone de tres métodos m1, m2 y m3. El programador decide que deben ejecutarse en exclusión mutua, y para ello dispone todo el código de los métodos m1 y m3 dentro de un bloque synchronized(this). Para el tercer método, utiliza un objeto de la clase **Semaphore que inicializa a cero,** y engloba todo el código del mismo bajo el par de instrucciones acquire() y reléase(). La sección crítica en todos los casos implica incrementar una variable de tipo int que inicialmente vale 3. Tres hebras externas A, B y C activadas dentro de una co-rutinas ejecutan lo siguiente: A.m1, C.m3 y B.m2. Terminada la co-rutina, el programa principal imprime el valor de la variable y este es:

a) 3, ya que las tres hebras quedan bloqueadas y no lo modifican.
b) 6, pues las tres hebras hace su incremento de modo seguro.
c) El programa realmente no imprime nada porque queda bloqueado a causa de las hebras
d) 5, porque dos hebras hacen su incremento y otra no, ya que queda bloqueada.

24- [FEB 2018] En Java, utilizando cerrojos de la clase ReentrantLock y variables Condition es posible disponer

a) De señalización SC y SX
b) De señalización SA, SC y SX
c) De señalización SC y SA
d) Ninguna de las anteriores es correcta

25- [FEB 2018] Una hebra de C++

a) Únicamente puede recibir su código mediante un puntero a función
b) Tras ser instanciada mediante el constructor de clase, debe ser lanzada explícitamente por programa
c) Puede estar sincronizada con el programa principal o con otras mediante el método join()
d) Todas son correctas

26- [FEB 2018] Desarrollando el framework RMI, el binario **rmiregistry**:

a) Permite registrar diferentes objetos de diferentes clases
b) Permite registrar diferentes instancias de un mismo objeto
c) Efectúa funciones de DNS mediante clientes que hace dynamic-bindig con él
d) Todo lo anterior es cierto

27- [FEB 2018] Se desea implementar un sistema de control de una válvula de presión en una central térmica, y usted determina emplear para ello el API de JRTS. Dado lo anterior, estima que necesita controlar de forma periódica la presión en la válvula, mientras que ocasionalmente inyecta fuel-oil en las turbinas de generación. En consecuencia, deberá utilizar

a) Hebras de T.R periódicas y gestores de eventos asíncronos ligados a hebras T.R esporádica
b) Hebras de T.R esporádicas y gestores de eventos asíncronos ligados a hebras de T.R periódicas
c) Únicamente gestores de eventos asíncronos
d) Ninguna de las anteriores es correcta, y basta utilizar hilos de T.R

28- [FEB 2018] Es conocido que la anidación de bloqueos en lenguaje Java sobre dos recursos compartidos puede llevar a dos hebras, en el peor de los casos, a una situación de deadlock. Para evitar lo anterior podemos

a) Evitar el anidamiento de bloqueos mediante un monitor
b) No utilizar recursos compartidos
c) Utilizar el API de alto nivel del lenguaje, que ofrece en tiempo de compilación la capacidad para detectar y eliminar estas situaciones
d) Reducir a cero la probabilidad de aquellos flujos de ejecución que conlleven al interbloqueo, cambiando la arquitectura de la codificación / verificar de manera formal las propiedades de corrección de nuestro código

29- [FEB 2018] Cuando se utiliza programación CUDA, el programa principal que lanza el kernel CUDA:

a) Está sincronizado con parte de los kernels, pero no con todos
b) Puede ejecutar código paralelo y actuar como coprocesador de la GPU
c) Puede tener hebras ejecutándose sobre la CPU, pero deben estar en exclusión mutua con las hebras que ejecutan los kernels de la GPU
d) Nada de lo anterior es cierto

¿Qué es un kernel?: Instrucciones a aplicar sobre unos datos dados (stream)

30- **[FEB 2018]** El uso de variables volatile en Java

a) Garantiza al programador que los datos que albergan son accedidos de forma segura frente a hebras concurrentes
b) Elimina las cache temporales de esa variables asociadas al motor de ejecución que las hebras utilizan
c) Permite garantizar código parcialmente correcto, pero no totalmente correcto.
d) Ninguna de las anteriores es cierta

31- **[FEB 2018]** El resultado de paralelizar una aplicación sobre una máquina de dos cores ofrece un **speedup** de 0.75. Esto significa que

a) El código original es inherentemente paralelo
b) El coeficiente de bloqueo de la aplicación está próximo a 1
c) Su diseño paralelo es ineficiente
d) Ninguna de las anteriores es correcta

32- **[FEB 2018]** Cuando se utiliza STM en Java sobre Clojure:

a) El delimitador sintáctico dosync gestiona el acceso a regiones criticas bajo transacciones
b) Los interbloqueos siguen sin poder evitarse
c) Puede haber inconsistencias de la información
d) Es necesario utilizar, también a nivel sintáctico, el delimitador LockingTransaction.runInTransaction()

33- **[FEB 2018]** En el lenguaje C++ el desarrollo de un monitor supone:

a) Disponer de una única cola de espera por condición para las tareas
b) Una semántica de señalización de tipo SX
c) Es imposible usarlos. C++ no lo permite
d) Ninguna de las anteriores son ciertas

34- **[FEB 2018**] La implementación de la primitiva teórica de región critica en C++

a) Es imposible
b) Es posible utilizando un objeto auxiliar que actúe como cerrojo
c) Es posible encerrando la región entre lock.lock() y lock.unlock() donde lock es un objeto de la clase mutex
d) Ninguna de las anteriores son ciertas

35- **[FEB 2018]** El uso de ejecutores de pool de thread en Java

a) Optimiza el rendimiento para cualquier número de hebras
b) Permite al programador centrarse exclusivamente en las tareas
c) Imposibilita el uso de clases contenedoras autosincronizadas
d) Solo tiene sentido en problemas y tareas con C_b=0

37- **[FEB 2018]** La reentrancia para código concurrente seguro en el lenguaje C++:

a) Funciona exactamente igual que en Java con cualquier tipo de cerrojo que se escoja
b) Funciona únicamente escogiendo cerrojos de la clase mutex
c) No es posible disponer de ella.
d) Ninguna de las anteriores es cierta

38- **[FEB 2018]** Usted debe determinar el valor de C_b óptimo para una aplicación con paralelismo a nivel de hebras que acaba de escribir. Para ello:

a) Busca el mínimo de la curva Tiempo=f(Cb) mediante experimentación, lo determina de forma aproximada, y deduce a partir de aquí cuántas hebras necesita tener en su aplicación. Busca el mínimo de la curva Tiempo=f(Cb) mediante técnicas analíticas.
b) Busca el máximo de la curva Tiempo=f(Cb) mediante experimentación, y lo determina de forma aproximada.
c) Queda determinada por la tipología del problema
d) No es posible determina esto de forma adecuada sin un benchmarking profesional. Puesto que su código tiene latencias de E/S y de red, su experiencia como programador le dice que Cb=0.5 aproximadamente.

39- **[FEB 2018]** El uso de la interfaz Future junto con tareas Callable

a) Es igual que con la interfaz Runnable, cambiando el método run() por el método call().
b) Permite el acceso a los datos de retorno del método call() bajo bloqueos explícitos.
c) Permite el acceso a los datos de retorno del método call() bajo bloqueos implícitos.
d) Permite procesar los objetos que la implementan dándolas como parámetros a objetos de la clase Thread.

40- **[FEB 2018]** El anidamiento de múltiples operaciones wait(S) sobre diferentes semáforos realizadas por hebras diferentes:

a) Es algo que la especificación estándar del lenguaje Java no permite
b) Conlleva irrevocablemente a un deadlock, con independencia del lenguaje que soporte a los semáforos.
c) Es algo que la especificación estándar del lenguaje Java no permite pero la de tiempo real si.
d) Se puede hacer con Java, para ambas especificaciones, y con cualquier lenguaje que soporte semáforos siempre que respeten la invariancia de los mismos.

41- **[JUN 2017]** Suponga que 500 hebras comparten el acceso a un objeto común, cada una a través de su propia referencia, y ejecutan el acceso a un método inc() de ese objeto que incrementa un contador que inicialmente vale cero. El programador protege el acceso haciendo que cada hebra ejecute la llamada a inc() envolviendo la misma en un bloque de la forma synchronized(this) {….miReferencia.inc();---}. El valor final del contador es:

a) 500
b) 499
c) Un número indeterminado entre 0 y 499
d) Un número indeterminado entre 1 y 500

42- [JUN 2017] En Java los cerrojos de la clase ReentrantLock se utilizan:

a) Para tener regiones criticas de grano muy grueso
b) Para no utilizar regiones synchronized
c) Para poder utilizar variables de condición si hacen falta
d) Para optimizar los accesos a los datos que protegen

43- **[JUN 2017]** Una hebra

a) Puede estar detenida en un punto de su código por un bloqueo de exclusión mutua, mientras ejecuta código en una zona diferente
b) Puede acceder a varios objetos diferentes de forma concurrente
c) Puede ser procesada a través de un ejecutor
d) Ninguna de las anteriores es correcta

44- **[JUN 2017]** Usted debe desarrollar una aplicación bajo el framework RMI que controla un sistema de reserva de billetes de avión, y sabe:

a) Que le sistema deberá ser concurrente
b) Que habrá que crear utilizar necesariamente control de exclusión mutua explícito en el lado del servidor
c) Que deberá crear una hebra de servicio por cada petición procedente de un cliente
d) Que utilizando generación dinámica de resguardos puede prescindir de utilizar un DNS local a la máquina que ejecuta el servidor.

45- **[JUN 2017]** Considere un objeto en Java que dispone de tres métodos m1, m2 y m3. El programador decide que deben ejecutarse en exclusión mutua, y para ello dispone todo el código de los métodos m1 y m3 dentro de un bloque synchronized(this). Para el tercer método, utiliza un objeto de la clase Semaphore que inicializa a 0, y engloba todo el código del mismo bajo el par de instrucciones acquaire() y reléase(). La sección crítica de todos los casos implica incremental una variable de tipo int que inicialmente vale 3. Tres hebras A, B y C activadas dentro de una co-rutina ejecutan lo siguiente: A.m1, C.m3 y B.m2. Terminada la co-rutina, el programa principal imprime el valor de la variable y este es:

a) 3, ya que las tres hebras quedan bloqueadas y no la modifican
b) 6, pues las tres hebras hacen su incremento de modo seguro.
c) 5, porque dos hebras hacen su incremento y otra no, ya que queda bloqueada.
d) El programa realmente no imprime nada porque queda bloqueado a causa de las hebras

46- **[JUN 2017]** Se desea implementar un sistema de control de una válvula de presión en una central térmica, y usted determina emplear para ello el API de JRTS. Dado lo anterior, estima que necesita controlar de forma periódica la presión en la válvula, mientras que ocasionalmente inyecta fuel-oil en las turbinas de generación. En consecuencia, deberá utilizar

a) Hebras de T.R. periódicas y gestores de eventos asíncronos ligados a hebras de T.R. esporádicas.
b) Hebras T.R. esporádicas y gestores de eventos asíncronos ligados a hebras de T.R. periódicas
c) Únicamente gestores de eventos asíncronos
d) Ninguna de las anteriores es correcta, y basta con utilizar hilos de T.R

47- **[JUN 2017]** Es conocido que la anidación de bloqueos en el lenguaje Java sobre dos recursos compartidos puede llevar a dos hebras, en el peor de los casos, a situación de deadlock. Para evitar lo anterior podemos

a) Evitar el anidamiento de bloqueos mediante un monitor.
b) Anidar transacciones
c) Verificar de manera formal las propiedades de corrección de nuestro código
d) Utilizar el API de alto nivel del lenguaje

48- **[JUN 2017]** Cuando se utiliza la programación CUDA, el programa principal que lanza el kernel CUDA:

a) Está sincronizado con parte de los kernels, pero no con todos
b) No puede ejecutar código paralelo y actuar como coprocesador de la GPU
c) Puede tener hebras sobre la CPU, pero deben estar en exclusión mutua con los kernels de la GPU
d) Nada de lo anterior es cierto

49- **[JUN 2017]** El uso de ejecutores en el lenguaje Java supone

a) Optimizar siempre el rendimiento de la aplicación
b) Procesar indistintamente tareas Callable y Runnable a través de ellos
c) No controlar la sincronización, pues son los ejecutores quienes controlan el ciclo de vida de las tareas.
d) Asegurar la exclusión mutua de las tareas sobre los datos compartidos.

50- **[JUN 2017]** El uso de variables volatile en el ámbito de la concurrencia

a) Nos garantiza que los datos con accedidos de forma seguro
b) Sirven solo para implementar algoritmos de control de exclusión mutua con espera ocupada
c) Únicamente tiene sentido cuando esas variables son estáticas
d) Ninguna de las anteriores es cierta

51- **[JUN 2017]** El resultado de paralelizar una aplicación sobre una máquina de ocho cores ofrece un speed-up de 2.75. Esto significa que

a) El código original es inherentemente paralelo
b) El coeficiente de bloqueo de esta aplicación está próximo a 0
c) No se han utilizado ejecutores en el procesamiento de las hebras paralelas
d) Ninguna de las anteriores es cierta

52- **[JUN 2017]** Cuando se utiliza STM en Java sobre Clojure:

a) Desaparece la necesidad de controlar la exclusión mutua
b) Los interbloqueos siguen sin poder evitarse
c) Tenemos garantizada la consistencia de la información
d) Tenemos escrituras concurrentes y lecturas no concurrentes

53- **[JUN 2017]** En el lenguaje C++ el desarrollo de un monitor exige:

a) El uso de sincronización call_once.
b) El uso exclusivo de recursive_mutex
c) Utilizar unique_lock y variables de condición si son necesarias
d) Ninguna de las anteriores es correcta.

54- **[JUN 2017]** La implementación del concepto teórico de región critica en Java

a) Exige que los métodos sean synchronized
b) Exige utilizar un objeto auxiliar que actúe como cerrojo
c) Exige acotar con synchronized(this) la región
d) Todas las anteriores son ciertas.

55- **[JUN 2017]** Los contenedores autosincronizados de Java

a) Optimizan el rendimiento de los productores-consumidores
b) Elevan el nivel de abstracción para el programador
c) Pueden manejarse desde tareas gestionadas por ejecutores
d) Todas las anteriores son cierta

56- [JUN 2017] Se desea paralelizar un algoritmo para el que se ha determinado un Cb=0, y que tiene que trabajar sobre una máquina de 8 cores lógicos, para procesar un vector de 10^6 componentes sin interdependencia entre ella. Con estos datos, usted decide:

a) Utilizar un contenedor sincronizado para optimizar el rendimiento
b) Utilizar 16 tareas con 16 sub-vectores disjuntos y procesarlas utilizando un objeto de la clase ThreadPoolExecutor con corePoolSize=4
c) Utilizar 16 tareas con 16 sub-vectores y procesarlas utilizando un ejecutor de tamaño fijo mediante newFixedThreadPool(16)
d) Ninguna de las anteriores es cierta

57- [Jun 2017] La reentrancia para código synchronized en el lenguaje Java

a) Tiene sentido cuando un objeto es un monitor
b) Funciona cuando el código synchronized reentrante es recursivo
c) Funciona cuando un objeto concreto tiene código synchronized reentrante y código no synchronized
d) Todas las anteriores son ciertas

58- [Jun 2017] Los hilos Daemon en Java

a) Ejecutan tareas de alto y bajo nivel que no terminan nunca
b) Tienen sentido en sistemas reactivos
c) Permanecen activos hasta que se apaga la maquina
d) Solo hay realmente un hilo daemon correspondiente al garbaje collector

59- [Jun 2018] Considere el nivel MIMD de la taxonomía de Flynn. Dicho nivel contempla:

a) Arquitectura de acoplamiento fuerte
b) Arquitectura de acoplamiento débil
c) Ambas
d) Arquitecturas SIMD mejoradas con memoria NUMA
e) Ninguna de las anteriores es cierta

60- [JUN2018] Usted dispone de un servidor remoto soportado mediante el framework RMI. Dicho servidor define entre otros, un recurso que debe gestionarse bajo control de exclusión mutua. Su equipo de programadores controla dicha gestión mediante un método void myControl () que protege el recurso mediante el API de la clase Semaphore. Sus programadores crean el semáforo necesario mediante la instrucción **Semaphore mySem = new Semaphore(0).** El objeto servidor se registra, y queda en espera de peticiones sobre el puerto 2001. Múltiples clientes situados en máquinas virtuales distintas piden a través sus referencias remotas la ejecución del métodomyControl(), y la arquitectura remota:

a) Procesa secuencialmente esas peticiones, obligada por el sistema de gestión de la concurrencia del framework RMI.
b) Procesa esas peticiones en paralelo, incluso en la zona de código protegida en exclusión mutua con el semáforo.
c) Sus programadores han desarrollado un código que no es totalmente correcto pero sí parcialmente, y usted los abronca.
d) Sus programadores han desarrollado un código que no es parcialmente correcto, y usted los despide.
e) La arquitectura no tiene problema alguno bajo los supuestos planteados, y funciona normalmente. Usted felicita a sus programadores.

61- [JUN2018] Usted desea implementar la primitiva teórica de control de la concurrencia conocida como región critica. Para lograrlo, decide que lo correcto es:

a) Englobar el código de la región con un cerrojo de clase ReentrantLock
b) Englobar el código de la región critica utilizando un objeto de la clase Semaphore y su API
c) Utilizar un objeto de clase AtomicInteger, ya que el recurso al fin y al cabo es un simple contador
d) Utilizar un monitor utilizando el API de alto nivel de Java, ya que es la construcción que en Java más se aproxima al concepto de región critica
e) Nada de lo anterior es cierto

62- [JUN2018] Cuando usted utiliza una implementación de un monitor con el API de alto nivel en Java

a) Sabe que la misma utiliza la señalización SU, para optimizar el rendimiento todo lo posible
b) Sabe que debe indicar al monitor mediante un método especifico el tipo de señalización a utilizar, para así adaptarlo a sus necesidades especificas
c) Usted utiliza el monitor, y no se preocupa de nada mas
d) Sabe que la señalización en este contexto es SA, y por tanto no hay señalización explicita en el monitor
e) Nada de lo anterior es cierto

63- [JUN2018] La arquitectura GPU en CUDA presenta una jerarquía de memoria de tres niveles porque

a) Era lo adecuado para mapear texturas en el ámbito de los videojuegos, donde tuvo su origen
b) Era obligatorio para un adecuado balance de carga de procesamiento entre la GPU y la CPU
c) Al programador en C, el modelo de memoria del lenguaje le obligo a ello
d) Porque pertenece al nivel MIMD de la taxonomía de Flynn
e) Porque permite un mayor nivel de paralelismo entre hebras CUDA

64- [JUN2018] Considere la especificación JRTS para implementar una dirección de tiro en tiempo real con seguimiento del blanco. El proceso que sigue al blanco hasta que se encuentra a distancia de tiro debe ser construido de manera que chequee la posición del mismo mediante radar cada diez nanosegundos. Cuando el blanco está a distancia de tiro (menos de cinco kilómetros), se activa otro proceso que lanza un misil de contramedidas. Con esta especificación usted sabe que necesita.

a) Utilizar dos hebras de clase RealTimeThread, una periódica y otra esporádica.
b) Utilizar dos hebras de clase RealTimeThread, una periódica y otra aperiódica.
c) Utilizar dos hebras de clase NoHeapRealTimeThread, una periódica y otra esporádica.
d) Utilizar dos hebras de clase NoHeapRealTimeThread, una periódica y otra activada por la anterior cuando la distancia es menor a 5 km.
e) Utilizar una hebra clase NoHeapRealTimeThread y un gestor de eventos asíncrono.

65- [JUN2018] En el lenguaje C++ el desarrollo de un monitor supone

a) Una semántica de señalización SU
b) Disponer de múltiples colas de espera por condición para las tareas obligatoriamente
c) Que es imposible usarlos, C++ no lo permite
d) Disponer de múltiples colas de espera por condición para las tareas, en su caso
e) Disponer de un único wait-set análogo al que existe en el API estándar de Java

66- [JUN2018] Se desea implementar una versión paralela del producto escalar de dos vectores de 8 x 10^6 números reales de doble precisión sobre un procesador de 4 cores. Con el objetivo de lograr el máximo speedup usted determina:

a) Utilizar 8x10^6 hebras y lograr speed-up hiperlineal
b) Utilizar 8x10^6 tareas Runnable y un ejecutor, lo cual mejora la opción anterior
c) Utilizar 8 tareas Runnable , un ejecutor y un contador compartido
d) Utilizar 8 hebras, 8 contadores parciales y un contador global
e) Utilizar 4 hebras, 4 contadores parciales y un contador global

67- [JUN2018] Se desea paralelizar un algoritmo para el que se ha determinado que Cb es 0,5 y que tiene que trabajar sobre una máquina con 8 cores lógicos, para procesar 60 tareas con apertura de sockets a servidor remoto sin interdependencia entre ellas. Con estos datos, usted decide:

a) utilizar un contenedor sincronizado de tareas para optimizar el rendimiento
b) utilizar 60 tareas y procesarlas utilizando un objeto de clase ThreadPoolExecutor con corePoolSize=4
c) utilizar 60 tareas procesarlas utilizando un ejecutor de tamaño fijo mediante newFixedThreadPool(16)
d) utilizar 60 tareas y procesarlas utilizando un objeto de clase ThreadPoolExecutor con corePoolSize=8
e) ninguna de las anteriores es cierta

68- [JUN2018] Se necesita procesar en paralelo sobre un procesador multicore un cubo de datos de 10^3 x 10^3 x 10^3 componentes con interdependencias conocidas entre datos vecinos (la vecindad es cúbica, de longitud uno y con condiciones de frontera nulas), para generar una nueva versión del cubo con datos actualizados resultado del procesamiento. Asuma que dispone de la memoria necesaria parametrizando la JVM mediante el flag adecuado. Para lograr el mejor rendimiento posible, usted decide:

a) Utilizar tantas tareas como cores tenga el procesador, y escribir los datos en el mismo cubo original
b) Utilizar tantas tareas como cores tengas el procesador, y escribir los datos en un segundo cubo, dado que dispone de memoria para ello
c) Utilizar tantas tareas como cores tenga el procesador, y escribir los datos en el mismo cubo original, con control de exclusión mutua
d) Utilizar tantas tareas como cores tenga el procesador, y organizar los datos mediante secciones transversales del cubo, que se insertan en un objeto de clase ArrayList
e) Ninguna de las anteriores es cierta

70- [JUN2018] El multiprocesamiento simétrico mediante particionamiento del dominio de datos

a) Es propio y limitado al lenguaje Java
b) Es un modelo de programación paralela
c) Solo tiene sentido en lenguajes de programación orientado a objetos
d) Es inaplicable con las arquitecturas de procesadores actuales. Está obsoleto
e) Tiene sentido en problema de Cb próximo a 1

71- [JUN2018] Considere una hipotética instrucción máquina llamada leerYAsignar que puede ser llamada desde su lenguaje de programación favorito, que admite como parámetro la dirección de memoria de una variable lógica que actúa como cerrojo, y que ejecuta bajo atomicidad las siguientes tareas, en el orden en que aparecen: **lee el anterior valor del cerrojo, pone el valor del cerrojo a true y devuelve el anterior valor del cerrojo**. A partir de aquí, se escribe el siguiente código concurrente:

```
var libre:boolean := false;
        n:integer := 1;
  process [P[i:1..n] {
        var i:integer i= 0;

                while(i < 10) {
                  while(leerYAsignar(libre)){
                        n:=n+1;
                        libre := false;
                    i:= i+1;
                   }
              }
        }
```

El valor final de la variable compartida n si se lanzan diez instancias de este proceso es:

a) 100
b) 1000
c) un número en el intervalo 0-1000
d) 10
e) es un código que no verifica la propiedad de seguridad.

72- [JUN2018] Cuando se utilizan transacciones en lugar de bloqueos de exclusión mutua

a) El rendimiento de la aplicación aumenta siempre
b) El rendimiento de la aplicación no tiene por qué aumentar
c) Pueden dar lugar a interbloqueos
d) No permite compartir recursos entre tareas concurrentes
e) No tienen por qué ejecutarse atómicamente

73- [JUN2018] El método start() de la clase Thread

a) Fuerza a que la hebra sobre la cual se invoca comience su ejecución de manera inmediata
b) Sitúa al objeto sobre el cual se invoca en el heap, proporciona una referencia al mismo, y comienza su ejecución de manera inmediata
c) Puede ser invocado sobre objetos que expresan concurrencia implementando la interfaz Runnable
d) Es necesario invocarlo inmediatamente después de enviar una tarea que expresa concurrencia implementando la interfaz Runnable a un ejecutor
e) Nada de lo anterior es cierto

74- [JUN2018] El uso de un contenedor de datos autosincronizado entre varias tareas concurrentes

a) No requiere que todas las tareas posean una referencia al mismo
b) Solo es posible cuando las tareas son procesadas mediante un ejecutor
c) Mejora el rendimiento siempre
d) Mejora el rendimiento en determinadas ocasiones
e) Nada de lo anterior es cierto

75- [JUN2018] La verificación de la propiedad de seguridad mediante las ecuaciones de Bernstein

a) Es adecuada para una demostración formal de la seguridad
b) Evita las escrituras concurrentes
c) Es inaplicable en la práctica, cuando trabajamos con código concurrente de cientos o miles de instrucciones no atómicas
d) Todas las anteriores son ciertas
e) Ninguna de las anteriores es cierta

Faltan: Sep2017 - Sep2018 - Feb,jun,sep 2016