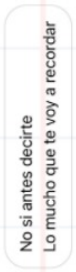
PCTR-TEST-1.pdf

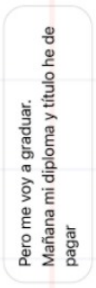
KologSensei

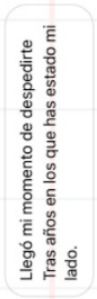
Programación Concurrente y de Tiempo Real

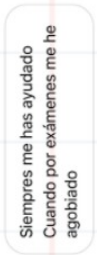
3º Grado en Ingeniería Informática

Escuela Superior de Ingeniería
Universidad de Cádiz









PREGUNTAS TEORÍA PCTR

Preguntas	Examen
1-20	Junio 2020
21-40	Septiembre 2020
41-60	Febrero 2021
61-100	Exámenes Wuolah
101-120	Febrero 2020
121-140	Junio 2021
141-160	Septiembre 2021

PREGUNTAS TEORÍA PCTR

1 Se programa en Java una clase M que consta de 5 métodos de los cuales 3 de ellos están sincronizados mediante un cerrojo de clase *ReentrantLock*, y dos no. La clase define un conjunto de atributos. A continuación se escribe un programa que instancia un objeto O de clase M, y crea hebras que comparten el acceso a O mediante variables de referencia. En esas condiciones…

a) El objeto O implementa de forma teórica el concepto de monitor.
b) Existe exclusión mutua sobre los datos que el objeto O encapsula.
c) Es posible tener en tiempo t múltiples hebras accediendo al objeto O.
d) No existe reentrancia para aquellas hebras que acceden al objeto O a través de métodos que están sincronizados.

2 El uso de variables *volatile* en el ámbito de la concurrencia...

a) Nos garantiza que los datos almacenados en ellas son accedidos de forma segura.
b) Permite garantizar código parcialmente correcto, pero no totalmente correcto.
c) Sirven para optimizar el rendimiento de la aplicación.
d) Fuerza a que la lectura / escritura de estas variables se haga en memoria principal.

3 Se desea paralelizar un algoritmo para el que se ha determinado un coeficiente de bloqueo C_b=0, y que tiene que trabajar sobre una máquina con 8 cores lógicos, para procesar un vector de 10^6 componentes sin interdependencia conocidas entre ellas. Con estos datos, usted decide:

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

4 Los monitores escritos en lenguaje C++…

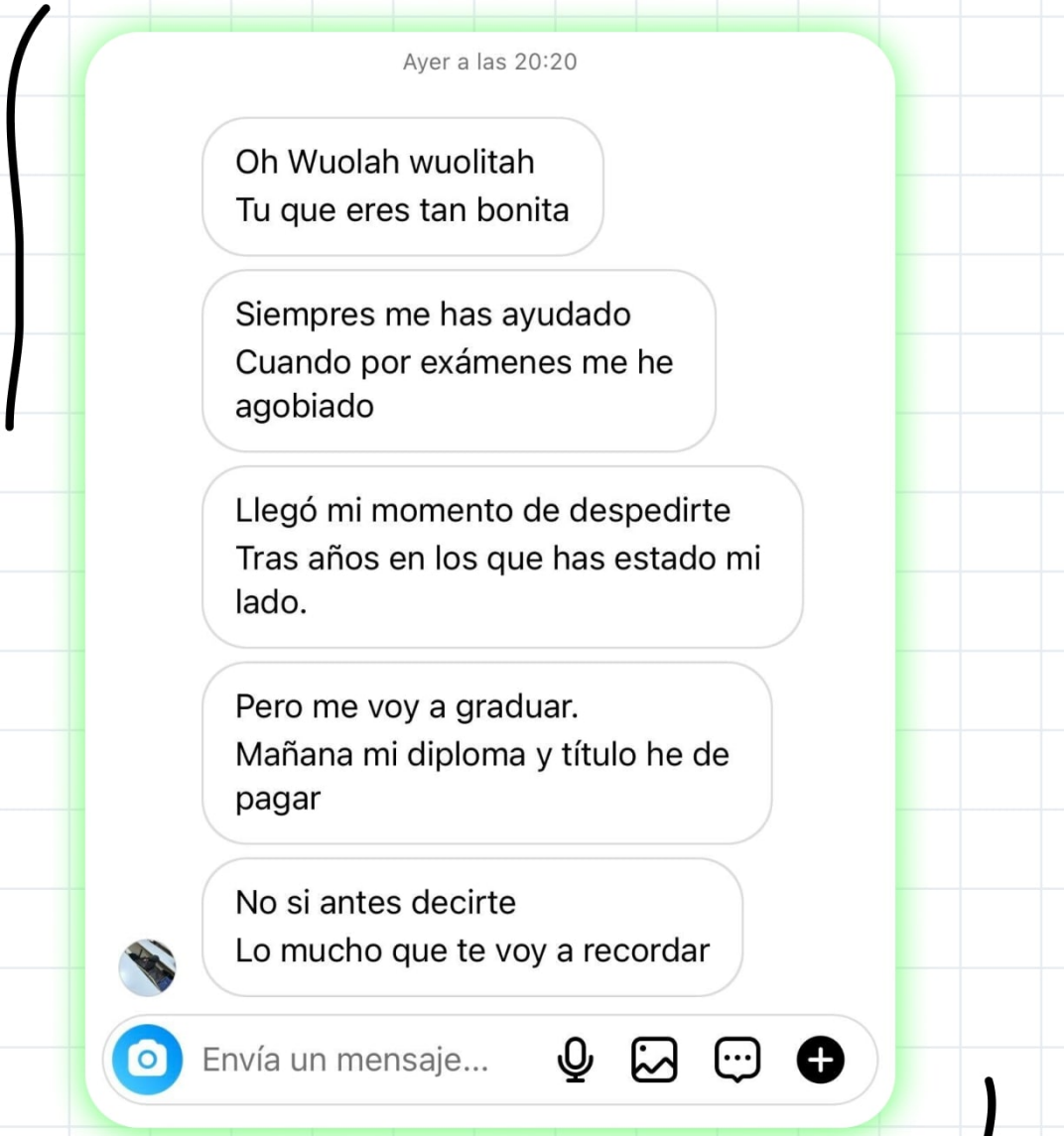
a) Permiten varias colas de espera por condición.
b) Permiten únicamente una cola de espera por condición, con señalización SU.
c) Funcionan únicamente para proteger recursos compartidos bajo control de exclusión mutua, igual que una región crítica.
d) Funcionan igual que los monitores del lenguaje Python.

5 El multiprocesamiento simétrico que utiliza paralelismo mediante división del dominio de datos entre las tareas...

a) Es propio de y limitado al lenguaje Java.
b) Es un modelo de programación paralela.
c) Solo tiene sentido en lenguajes de programación sin orientación a objetos.
d) Tiene sentido en problemas con coeficiente de bloqueo con un valor próximo a la unidad.

Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

6 Se desea paralelizar un algoritmo para el que se ha determinado que C_b=0.8 y que tiene que trabajar sobre una máquina con 8 cores lógicos, para procesar 60 tareas con apertura de sockets a un servidor remoto y posterior descarga de datos, sin interdependencias conocidas entre ellas. Con estos datos, usted decide:

a) Utilizar un contenedor sincronizado de tareas para optimizar el rendimiento.
b) Utilizar 60 tareas y procesarlas empleando un objeto de clase *ThreadPoolExecutor* con *corePoolSize* = 4.
c) Utilizar 60 tareas y procesarlas empleando un objeto de clase *ThreadPoolExecutor* con *corePoolSize* = 40.
d) Utilizar 60 tareas y procesarlas empleando un objeto de clase *ThreadPoolExecutor* con *corePoolSize* = 16.

7 Considere un problema de naturaleza numérica que se ha resuelto mediante procesamiento paralelo utilizando un procesador multicore de 16 cores, y donde no se han identificado latencias conocidas; se sabe que la paralelización ha sido bien realizada, y las pruebas realizadas para un rango de tareas paralelas de 2 a 32 ilustran un speedup máximo de 0.98 para 2 tareas. Usted deduce:

a) Que el coeficiente de bloqueo del problema está próximo a 1.
b) Que el problema es inherentemente no paralelizable.
c) Que el análisis debería haber probado un número de tareas superior a 32.
d) Nada de lo anterior es cierto.

8 La planificación mediante ejecutivo cíclico en un sistema de tiempo real:

a) Se realiza on-line de forma dinámica durante la ejecución del sistema.
b) Es completamente segura frente a la concurrencia sin necesidad de programar un control explícito de exclusión mutua.
c) Tiene un hiperperiodo Tm = mcd (T1, T2, ..., Tm) Tm = mcd (T1, T2, ..., Tm).
d) Aprovecha todos los cores disponibles de la plataforma.

9 Desde el punto de vista de las primitivas teóricas de control de la concurrencia, una región crítica…

a) Puede simularse únicamente con semáforos.
b) Puede simularse con semáforos y también con monitores.
c) Puede simularse únicamente con monitores.
d) Las primitivas de control teórico de la concurrencia no pueden simularse entre sí.

10 Cuando se utiliza el modelo de memoria transaccional software en Java sobre Clojure…

a) El delimitador sintáctico dosync gestiona el acceso a regiones críticas bajo transacciones gestionadas por el manejador de transacciones.
b) Los interbloqueos entre tareas concurrentes siguen sin poder evitarse.
c) Puede haber inconsistencias de los datos compartidos entre tareas que están protegidos bajo transacciones.
d) Es necesario usar a nivel sintáctico, el delimitador *LockingTransaction.runInTransaction()*.

11 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(2)*. 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 método *myControl()*, y la arquitectura remota:

a) Procesa secuencialmente esas peticiones, obligada por el sistema de gestión de la concurrencia del framework RMI.
b) Sus programadores han desarrollado un código que es seguro pero carece de vivacidad, y usted los abronca.
c) Sus programadores han desarrollado un código que no es seguro, y usted los despide.
d) La arquitectura no tiene problema alguno bajo los supuestos planteados, y funciona normalmente preservando la integridad de los datos. Usted felicita a sus programadores.

12 El uso de un enfoque paralelo en la resolución de un problema da lugar a:

a) Una mejora del rendimiento de la solución paralela frente a la secuencial, siempre.
b) No mejora nunca el rendimiento
c) Una mejora del rendimiento dependiente de la naturaleza del problema.
d) Una mejora del rendimiento dependiente, exclusivamente, del número de cores de la máquina.

13 Cuando se programa el control de la exclusión mutua con las primitivas disponibles para ello en los lenguajes de programación Java, C++ y Python, la característica de "reentrancia":

a) Es inherente a todas las primitivas para los tres lenguajes.
b) No es necesariamente inherente a todas las primitivas para los tres lenguajes.
c) Depende de cómo estén soportados las hebras en los diferentes lenguajes.
d) Depende del modelo de memoria que cada lenguaje implementa.

14 El lenguaje Java soporta la implementación del concepto teórico de monitor mediante el los API estándar y de alto nivel. A partir de aquí, considere un problema concurrente resuelto mediante un monitor teórico que requiere la presencia de cinco variables de condición. A continuación se escriben implementaciones con Java de este monitor con ambos API, y para ambas, el número de tareas que utilizan ambos implementaciones es elevado. Entonces:

a) La versión del monitor con el API de alto nivel resultará menos eficiente que la versión con el API estándar.
b) Ambas versiones tendrá una eficiencia equivalente.
c) Dado que con ambas API, en tiempo solo hay una hebra activa en el monitor, no tiene sentido especular sobre el rendimiento.
d) La versión del monitor con el API de alto nivel resultará más eficiente que la versión con el API estándar.

15 La técnica de paralelismo de datos mediante división del dominio de datos entre múltiples tareas paralelas puede desarrollarse adecuadamente

a) Con Python utilizando tareas soportadas mediante objetos de la clase Thread.
b) Únicamente con Python cuando se dispone de una GPU.
c) Con los lenguajes Java, C++, Python pero no con el API MPJ-Express para Java.
d) Con los lenguajes Java, C++, Python y el API MPJ-Express para Java.

16 Considere un sistema que controla el acceso concurrente a información compartida entre tareas. Sabemos que el acceso a dicha información implica un número de operaciones de escrituras concurrentes muy alto frente al número de lecturas concurrentes, e igualmente sabemos que la carga de tareas será elevada. En esas condiciones, usted como diseñador de una solución eficiente para el sistema

a) Descarta de inmediato una solución basada en memoria transaccional software.
b) Considera que merece la pena implementar soluciones con memoria transaccional software y bloqueos estándares y desarrollar un *benchmarking* para estimar cuál de ellas le ofrece mejor rendimiento.
c) Dado el desbalance entre el número de operaciones de lectura y escritura, estima que la solución basada en memoria transaccional software y opta por ella.
d) Considera que una solución con memoria transaccional que utilice Java sobre Clojure es la mejor.

17 Para programar una solución a un problema con coeficiente de bloqueo Cb=0 utilizando el lenguaje Python usted escogería una aproximación

a) De paralelismo con procesos, compartiendo memoria.
b) De paralelismo con procesos, comunicándolos mediante mensajes.
c) De paralelismo con hebras, compartiendo memoria y utilizando bloqueos de exclusión mutua cuando sea necesario.
d) El Global Interpreter Lock (GIL) impide que Python desarrolle un paralelismo real de ninguna forma.

18 Una hebra soportada en Java mediante una lambda-expresión

a) Puede estar detenida en un punto de su código por un bloqueo de exclusión mutua, mientras ejecuta código de una zona diferente.
b) Puede acceder a varios objetos diferentes de forma concurrente.
c) Puede ser procesada a través de un ejecutor.
d) Tiene un ciclo de vida distinto a una hebra soportada mediante Runnable.

19 Usted debe desarrollar una aplicación bajo el framework RMI que controla un sistema de reserva de billetes de avión, y sabe:

a) Que puede situar DNS y objetos servidores en máquinas diferentes.
b) Que habrá que 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.

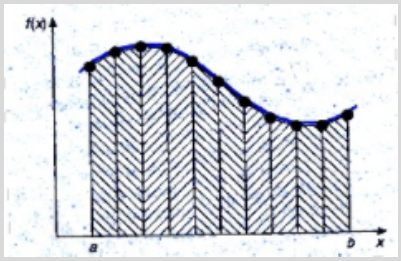
20 El uso de la interfaz Future junto con tareas Callable:

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, que el programador debe definir escogiendo alguna técnica de control adecuada.
c) Permite procesar a esas tareas dándolas como parámetros a objetos de la clase Thread.
d) **Permite el acceso a los datos de retorno del método call() de manera segura, cuando la elección y gestión de las estructuras de datos se hace adecuadamente.**

21 Se desea desarrollar un protocolo que permita, dado un monitor escrito en C++ que soporta inter-hebras mediante múltiples colas de condición, obtener un monitor semánticamente equivalente utilizando el API estándar de Java. Usted considera que dicho protocolo debe:

a) Codificar un método synchronized en Java por cada método del monitor en C++.
b) Codificar un método synchronized en Java por cada método del monitor Java, y codificar la sincronización en el monitor Java, declarando una variable *Condition* por cada *variable_condition* que aparece en el monitor con C++
c) No es posible construir utilizando el API estándar de Java un monitor semánticamente equivalente a otro escrito en C++, ya que al tener un único wait-set, es imposible codificar la sincronización que el monitor en C++ ofrece mediante múltiples variables de condición.
d) **Ninguna de las alternativas ofrecidas en las respuestas anteriores es correcta.**

22 Considere una función real de variable real f(x). Se desea aproximar la integral de la función en el intervalo [a, b] de forma numérica utilizando la regla del trapecio, que divide el intervalo en una partición de k subintervalos, y aproxima la integral mediante la suma de la superficie de los k trapecios resultantes, según ilustra la siguiente figura. Para resolver este problema de forma paralela sobre una máquina octa-core sin hyperthreading, utilizando el lenguaje Java, usted estima que:



a) Necesita procesar cuatro tareas Runnable almacenando las superficies de cada trapecio en posiciones distintas de un array con necesidades obligadas de control de exclusión mutua.
b) Necesita procesar ocho hebras con lambda-expresiones, almacenando (y sumando) la superficie de cada trapecio en una variable “double” estática llamada “integral”, sin necesidades obligadas de control de exclusión mutua.
c) **Necesita un array de tipo “double” modelado a “Future”, soportando a ocho tareas mediante la interfaz Callable, retornando a ese array la superficie de cada trapecio a través del retorno del método call().**
d) Que es mejor utilizar un esquema de paso de mensajes utilizando MPJ-Express, de forma que cada tarea reciba un elemento de la partición, calcule la superficie de un trapecio, y la retorne al programa principal para que la sume con las demás.

23 El API MPJ-Express para paso de mensajes en lenguaje Java logra implementar comunicación uno a muchos mediante:

a) Direccionamiento directo con la instrucción MPI.COMM_WORLD.Send.
b) Direccionamiento directo con la instrucción MPI.COMM_WORLD.Ssend.
c) Direccionamiento directo con la instrucción MPI.COMM_WORLD.Bcast.
d) No es posible efectuar la difusión uno a muchos con Java y MPJ-Express.

24 Considere un problema orientado a la detección de patrones en secuencias de ADN, y su resolución distribuida sobre una arquitectura de tipo cluster de procesadores, que dispone de 40 nodos de procesamiento comunicados por una red dedicada que presenta cierta latencia en la transferencia de datos. Cada nodo dispone de 8 cores de ejecución y una memoria dedicada. En este contexto usted decide que la solución al problema utilizando el lenguaje Java...

a) Necesita usar 320 tareas y procesarlas a través de un ejecutor..
b) Necesita determinar el coeficiente C_b y determinar el número de tareas necesarias aplicando la ecuación de Subramanian.
c) Necesita utilizar la extensión de Java MPJ-Express en un esquema de tipo master-slave, diseminando la detección de secuencias desde una tarea master a tareas slaves situadas en los nodos.
d) Necesita utilizar paralelismo por división del dominio de datos entre 40 tareas, cada una situada en un nodo, como medio de lograr speedups superiores a 20.

25 El lenguaje C puede disponer de semáforos mediante el uso conjugado de las bibliotecas ipc.h y sem.h. Con ellas, el programador puede utilizar semáforos...

a) Con una semántica equivalente a la proporcionada por los semáforos teóricos.
b) Con una semántica que restringe a la proporcionada por los semáforos teóricos.
c) Con una semántica que amplía a la proporcionada por los semáforos teóricos.
d) Con una semántica equivalente a la de los semáforos de Java.

26 La planificación en la especificación a tiempo real de Java (JRTS) se construye...

a) Mediante sobreescritura de los métodos de gestión de prioridades de Java.
b) Mediante herencia de la clase Thread, incorporando nuevas subclases que dan soporte a gestión de prioridades que JRTS necesita.
c) Mediante la definición de una nueva jerarquía de clases de JRTS que da soporte a la gestión de prioridades para tiempo real.
d) Mediante la extensión del modelo de prioridades de Java estándar a un nuevo modelo que permite instanciar ejecutivos cíclicos para controlar la planificación.

27 Las ecuaciones de Bernstein son una herramienta de análisis matemático aplicable a programas concurrentes que...

a) Son de aplicación en sistemas concurrentes sin memoria común
b) Son aplicables en concurrencia con Java y MPJ-Express
c) Son aplicables con Java, C++ y Python con hebras
d) Son aplicables con Java, C++ y Python con procesos.

28 Utilizando el lenguaje Python se implementa una solución paralela para el producto de matrices de alta dimensión de números reales. La solución se ejecuta sobre una arquitectura de dos procesadores con un banco de memoria común de 128 GB de memoria, y un total, en conjunto, de 16 cores. Su equipo de programadores implementa la solución y efectúa una batería de pruebas. Le remiten un corto informe donde le indican que las pruebas muestran un speedup medio de 0,85. Usted deduce que...

a) Su equipo de programadores ha efectuado una implementación de la solución utilizando paso de mensajes con procesos.
b) Su equipo de programadores ha efectuado una implementación de la solución utilizando hebras, pero no ha considerado el impacto del Global Interpreter Lock sobre el rendimiento.
c) Su equipo de programadores ha efectuado una implementación de la solución utilizando paso de mensajes con procesos, pero no ha considerado las latencias derivadas de las operaciones de envío y recepción de mensajes.
d) Su equipo de programadores ha efectuado una implementación de la solución utilizando paso de mensajes con procesos y hebras, en un enfoque de paralelismo mixto.

29 Considere el API de alto nivel para control de la concurrencia en lenguaje Java y, dentro del mismo, el sub-API constituido por la clase Semaphore, que ofrece la implementación que este lenguaje hace del concepto teórico de semáforo. Este sub-API...

a) Ofrece exactamente la misma semántica que la definida por las operaciones wait(S) y signal(S) del concepto teórico de semáforo.
b) Ofrece una semántica que es un superconjunto que engloba y amplía a la definida por las operaciones wait(S) y signal(S) del concepto teórico de semáforo.
c) Ofrece una semántica que es un subconjunto que reduce a la definida por las operaciones wait(S) y signal(S) del concepto teórico de semáforo, ya que en Java no se necesita contemplar esa semántica con carácter íntegro y global.
d) Ofrece la garantía de que cualquier solución que se programe utilizándolo está necesariamente libre de “deadlocks”.

30 Se desea programar el problema del productor-consumidor con el API de alto nivel para control de la concurrencia en el lenguaje Java. Para ello, se escribe una clase C de Java que encapsula los recursos necesarios (array de datos y punteros de inserción y de extracción: In_ptr y Out_ptr), con el objetivo de poder instanciar a partir de C tantos monitores como se desee. Para ello, su equipo de programadores...

a) Debe escribir una clase con cerrojos de clase ReentrantLock y condiciones Condition, e incorporar la sincronización estándar necesaria para resolver el problema.
b) Debe escribir una clase con cerrojos de clase ReentrantLock y condiciones Condition, hacer estáticos los recursos encapsulados, e incorporar la sincronización estándar necesaria para resolver el problema.
c) Debe escribir una clase cuyos métodos estén cualificados como “synchronized”, definir condiciones Condition, e incorporar la sincronización estándar necesaria para resolver el problema.
d) Debe escribir una clase utilizando cerrojos ReentrantLock, utilizar el “wait-set” que los objetos Java soportan por defecto, e incorporar la sincronización estándar necesaria para resolver el problema.

31 Considere el siguiente código concurrente escrito en Python. Indique el comportamiento que tiene en tiempo de ejecución, si se proporciona 1000 como valor de entrada para la variable niter:

```
from threading import Thread
from threading import Lock
import time
klock       = Lock()
shared_cont = 0
niter       = int(input('iterations?'))
lock        = Lock()
class myThread(Thread):
        def init (self):
                Thread.init(self)
                self.niter=niter;
        def run(self):
                global shared_cont
                klock.acquire()
                for cont in range(niter):
                  lock.acquire ()
                  shared_cont-=1
                  lock.release()
def main():
    hebra1=myThread()
    hebra2=myThread()
    start=time.time()
    klock.acquire()
    hebra1.start()
    hebra2.start()
    hebra1.join()
    hebra2.join()
    end=time.time()-start
    print('tiempo: ', end)
    print(shared_cont)

if __name__ == '__main__':
    main()
```

a) Imprime -1000 y el tiempo de ejecución
b) Imprime -2000 y el tiempo de ejecución
c) Es un código que carece de vivacidad
d) Es un código que carece de seguridad y de vivacidad.

32 Los algoritmos de control de la exclusión mutua con variables compartidas y espera ocupada conforman un conjunto de técnicas para control de la concurrencia que son...

a) Altamente transportables entre arquitecturas diferentes.
b) Altamente flexibles para acomodar la gestión de un número de tareas concurrentes creciente en tiempo de ejecución.
c) Aplicables para programar concurrencia en sistemas basados en memoria compartida y en paso de mensajes.
d) Escasamente eficientes cuando el número de tareas que se programan en un sistema con memoria común es muy elevado, y se utiliza creación dinámica de tareas concurrentes.

33 Considere el framework RMI de Java para invocación de métodos pertenecientes a objetos situados remotamente, y donde es necesaria la existencia de un *middleware* (capas de *stub*). Las funciones de este *middleware* son...

a) Efectuar directamente la interfaz entre la aplicación Java y la tarjeta de red.
b) Efectuar la serialización de los parámetros que se transfieren, y gestionar la espera de los códigos que se comunican hasta que reciben los datos que necesitan.
c) Efectuar la serialización de los parámetros que se transfieren, y localizar al objeto servidor previa consulta al DNS soportado por *rmiregistry*.
d) Efectuar la serialización de los parámetros que se transfieren, y registrar el objeto servidor en el DNS soportado por *rmiregistry*.

34 Considere un lenguaje hipotético que implementa el modelo teórico de semáforo, conservando íntegramente la semántica de dicho modelo, mediante un API de operaciones await() y signal(), que funcionan como wait y signal. Con dicho lenguaje se escribe el problema concurrente siguiente. Indique el valor final de la variable n que produciría el programa, si se dispone de un programa principal en memoria que crea y ejecuta respectivamente una hebra de clase A, otra de clase B y otra de clase C:

```
static int n = 0;
static Semaphore s1 = new Semaphore(0);
static Semaphore s2 = new Semaphore(1);
static Semaphore s3 = new Semaphore(0);
```

```
static class A
  extends Thread {
  public void run() {
    s1.await();
    s2.await();
    n = 2 * n;
    s2.signal();
  }
}
```

```
static class B
  extends Thread {
  public void run() {
    s3.await();
    n = n * n;
    s2.signal();
  }
}
```

```
static class C
  extends Thread {
  public void run() {
    s2.await();
    n = n + 2;
    s1.signal();
    s3.signal();
  }
}
```

a) 2
b) 2, 4 o 8
c) 8
d) 4 o 8

35 La memoria transaccional software es una herramienta que permite el acceso de múltiples tareas concurrentes a datos compartidos, gestionándolos dentro del ámbito de una transacción. Su uso permite a los programadores garantizar códigos concurrentes que son...

a) Vivaces
b) Vivaces y seguros
c) Seguros
d) Vivaces, pero no seguros.

36 Utilizando el lenguaje Python es posible comunicar a los procesos concurrentes o paralelos utilizando el concepto de pipe o cauce. El uso de estos cauces implica que:

a) Podemos dividir un dominio de datos común entre los procesos concurrentes o paralelos.
b) Estamos utilizando paso de mensajes y direccionamiento directo entre los procesos concurrentes o paralelos.
c) Estamos utilizando paso de mensajes y direccionamiento indirecto entre los procesos concurrentes o paralelos.
d) Ninguna de las anteriores es correcta.

37 Considere un monitor escrito en lenguaje Java mediante el API de alto nivel que utiliza un único cerrojo de clase ReentrantLock para acotar íntegramente el par de métodos lock() y unlock() el código de tres de los seis métodos que el monitor posee. En estas condiciones, ese monitor es utilizado por un conjunto de 20 hebras concurrentes implementadas mediante la interfaz Runnable, 5 de las cuales acceden al monitor a través de los tres métodos acotados mediante el cerrojo, y las 15 restantes a través de los métodos restantes. Usted considera que:

a) Establece un orden total sobre las 20 hebras concurrentes que acceden al monitor.
b) Establece un orden parcial sobre las 5 hebras que acceden al monitor a través de los métodos acotados por el cerrojo.
c) No establece ningún tipo de orden sobre las hebras concurrentes que acceden al monitor.
d) Establece dos ordenes parciales, uno sobre el conjunto de cinco hebras, y otro sobre las quince hebras restantes.

38 Considere una solución paralela a un problema cuya naturaleza es de computación numérica, que ha utilizado paralelismo mediante división del dominio de datos entre tareas paralelas. Durante el análisis del problema, usted encuentra que una fracción del 50% del código de cada tarea debe ejecutarse secuencialmente debido a dependencias de datos. Sus desarrolladores le informan de que tras una batería de pruebas con la misma nube de datos sobre una arquitectura de 16 cores obtienen un speedup de 15,68. Con esos datos, usted decide que:

a) Sus desarrolladores han implementado correctamente la solución paralela analizada por usted.
b) Sus desarrolladores han controlado adecuadamente la ejecución secuencial de las fracciones de código que así lo requieren.
c) Sus desarrolladores han efectuado una mala implementación de la toma de tiempos y/o del cálculo del speedup en el programa.
d) Ninguna de las anteriores es correcta.

39 Cuando se programa paralelismo de datos utilizando el lenguaje Python con división automática del dominio de datos entre hebras concurrentes, para un problema de $C_b = 0$ del que se sabe mediante un análisis previo que admite una paralelización eficiente...

a) Es posible lograr speedups mayores a la unidad para dos o más hebras.
b) Es posible lograr speedups mayores a la unidad para dos o más hebras, pero únicamente cuando esas hebras se procesan a través de un ejecutor de pool de threads.
c) Es posible lograr speedups mayores a la unidad para dos o más hebras, siempre que determinemos el número de hebras aplicando la ecuación de Subramanian
d) Ninguna de las anteriores es correcta.

40 Considere el siguiente código escrito en lenguaje Java, e indique el comportamiento que tendrá en tiempo de ejecución:

```
public class X extends Thread{
    Object k;
    public X(Object l){k=l;}
    public void run(){
        synchronized(k){
                try{k.wait();}catch (InterruptedException e){}
                System.out.println(this.getName());
        }
    }

   public void x(){synchronized (k){k.notify();}}
   public void y(){synchronized (k){x();}}

   public static void main(String[] args)
      throws InterruptedException{
        Object c = new Object();
        X [] t = new X[10];
        for(int i=0; i<10;i++)
          {t[i]=new X(c); t[i].start();}
        t[5].x();
        Thread el = currentThread();
        el.sleep(2000);
        for(int i=0; i<10;i++)
          t[i].join();
        System.out.println("Fin...");
    }
}
```

a) El programa se ejecuta, imprime “Thread-0”, después imprime “Fin…” y termina.
b) El programa se ejecuta, imprime “Thread-0” y se bloquea debido a que todas las hebras toman dos veces un bloqueo sobre el cerrojo k.
c) El programa se ejecuta, imprime “Thread-0” y se bloquea debido a la gestión del wait-set que se ha implementado, no terminando normalmente.
d) El programa se ejecuta e imprime lo siguiente, pudiendo variar el orden de la impresión que cada hebra efectúa, debido al entrelazado:

Thread-0
Thread-4
Thread-8
Thread-9
Thread-2
Thread-5
Thread-1
Thread-3
Thread-6
Fin...

41 En un sistema de tiempo real...

a) El tiempo de cómputo requerido por una tarea no puede superar nunca el plazo de respuesta máximo.
b) El tiempo de cómputo requerido por una tarea periódica será siempre igual o inferior al periodo de activación de la tarea.
c) Las tareas se ejecutan de forma secuencial y no es necesario controlar el acceso en exclusión mutua a los recursos compartidos.
d) Las aplicaciones se ejecutarán habitualmente más rápido que en otro sistema con las mismas prestaciones pero que no imponga ligadura una temporal sobre los procesos.

42 Considere desde un punto de vista teórico la primitiva de control de la concurrencia conocida como monitor, y suponga que para un determinado problema, las necesidades de sincronización del mismo requieren un sistema de colas de condición compuesto por n unidades. El monitor va a dar soporte a la política de señalización mediante el modelo de señales SU. En estas condiciones, el número total de colas de este monitor es:

a) igual a n+1
b) igual a n+2
c) igual a n+3
d) igual a 2: un wait-set, y una cola de condición que simula al sistema de n colas.

43 Cuando en un sistema se produce un interbloqueo activo, significa que...

a) Los procesos no están en la cola de planificación de corto plazo del sistema operativo, sino en una cola de procesos bloqueados y no intentan acceder a la CPU.
b) Los procesos quedan completamente bloqueados a la espera de que otro proceso los desbloquee, lo cual no podrá ocurrir.
c) Los procesos siguen dentro de la planificación del sistema operativo, y consumen ciclos de CPU.
d) Los procesos han intentado modificar una variable simultáneamente, por lo que el sistema queda a la espera de una acción que indique el orden de modificación.

44 Se desea programar el problema de los filósofos con el API de alto nivel para control de la concurrencia en el lenguaje Java. Para ello, se escribe una clase C de Java que encapsula los recursos necesarios (array de tenedores), con el objetivo de poder instanciar a partir de C tantos monitores como se desee, para versiones del problema donde el número de filósofos es variable. Para ello, su equipo de programadores...

a) Debe escribir una clase con métodos que utilicen un cerrojo de clase ReentrantLock común a todos ellos, y múltiples condiciones "Condition", e incorporar la sincronización estándar específica del problema.
b) Debe escribir una clase con métodos que utilicen un cerrojo de clase ReentrantLock propio y específico de cada método, y múltiples condiciones "Condition", e incorporar la sincronización estándar específica del problema.
c) Debe escribir una clase cuyos métodos estén cualificados como synchronized, definir condiciones "Condition", e incorporar la sincronización estándar específica del problema.
d) Todas las respuestas son correctas.

45 Suponga que está implementando el conocido problema del productor-consumidor en el lenguaje C++. ¿Por qué razón implementaría el proceso consumidor en una función anónima o lambda?

a) Porque todos los recursos que utilice dentro de este tipo de funciones se considerarán atómicos.
b) Porque es necesario garantizar que otras hebras no acceden a los recursos compartidos.
c) Porque este tipo de funciones son las encargadas de arrancar el objeto de la clase thread de forma automática
d) Ninguna de las otras respuestas es correcta.

46 Considere el siguiente programa escrito en Java. ¿Cuál es su comportamiento en tiempo de ejecución?

```
public class V extends Thread {
   Condition[] c;
   int i;
   ReentrantLock l;

   public V(int i){
     this.i = i;
     l = new ReentrantLock();
     c = new Condition[this.i+1];
     for(int k=0; k<c.length; k++) c[k] = l.newCondition();
   }

   public void run (){
     l.lock();
     try {
       c[this.i].await();
       System.out.println("Soy la hebra "+this.i);
     } catch (InterruptedException e){} finally {l.unlock();}
    }

   public void aux() throws InterruptedException {
     l.lock();
     try {c[this.i].signal();
     } finally{l.unlock();}
    }

  public static void main(String[] args) throws Exception{
      V[] t = new V[4];
      int s = 2;
      for(int i=0; i<t.length; i++) t[i]=new V(i);
      for(int i=0; i<t.length; i++) t[i].start();
      Thread.currentThread().sleep(1000);
      t[s].aux();
      for(int i=0; i<t.length; i++) t[i].join();
      System.out.println("Fin del programa...");
  }
 }
```

a) El programa efectúa cuatro impresiones de la forma “Soy la hebra n”, donde n es un número entre 1 y 4, a continuación imprime “Fin del programa…” y termina normalmente.
b) El programa efectúa varias impresiones de la forma “Soy la hebra n”, donde n es u número entre 0 y 3 y queda bloqueado.
c) El programa efectúa una impresión de la forma “Soy la hebra 2”, y termina normalmente imprimiendo “Fin del programa…”
d) Ninguna de las anteriores es correcta.

47 Considere una clase escrita en C++ en la cuál una parte de sus métodos están sincronizados utilizando un cerrojo de clase mutex, mientras que los métodos restantes están sincronizados utilizando un segundo cerrojo de clase recursive_mutex. Todos los métodos tienen acceso a todos los atributos de la clase y existen métodos de índole recursiva cuya exclusión mutua es controlada bien con mutex estándar, bien mediante mutex recursivo. Bajo las condiciones descritas se crea un objeto de la clase, y se da acceso al mismo a múltiples hebras concurrentes. Podemos entonces asegurar que...

a) El acceso al objeto es seguro frente a exclusión mutua y bloqueos.
b) El acceso al objeto es seguro frente a exclusión mutua, pero no frente a bloqueos.
c) El acceso al objeto no es seguro ni frente a exclusión mutua, ni frente a bloqueos.
d) El acceso al objeto es seguro frente a bloqueos, pero no frente a exclusión mutua.

48 Considere el siguiente programa escrito en Java utilizando paso de mensajes mediante MPJ-Express. Se utiliza el binario *mpjrun.bat -np 2 xYZ*. ¿Cuál es la salida impresa que produce?

```
import mpi.*;

public class xYZ{
public static void main(String args[]) throws Exception {

 MPI.Init(args);
 int rank = MPI.COMM_WORLD.Rank();
 int size = MPI.COMM_WORLD.Size();
 int emisor = 0; int receptor = 1;
 int tag = 100; int unitSize = 1;
 if(rank==emisor){
     int[] v1=new int[3];
     v1[0]=2;
     v1[1]=4;
     v1[2]=6;
     int bufer1[] = new int[3];
     for(int i=0; i<bufer1.length; i++)
         MPI.COMM_WORLD.Send(v1, i, unitSize, MPI.INT, receptor, tag+i);
     MPI.COMM_WORLD.Recv(v1, 0, unitSize, MPI.INT, receptor, 10);
     System.out.println(v1[0]);
 } else{
         int revbufer[] = new int[3];
         for(int i=0; i<revbufer.length; i++)
         MPI.COMM_WORLD.Recv(revbufer, i, unitSize, MPI.INT, emisor, tag+i);
         int []aux= new int[1]; aux[0]=1;
         for(int i=0; i<revbufer.length; i++)
           aux[0]=aux[0]*(revbufer[i]*2);
         MPI.COMM_WORLD.Send(aux, 0, unitSize, MPI.INT, emisor, 10);
   }
 MPI.Finalize();
 }
}
```

a) 48
b) 384
c) 128
d) 732

49 Para que un protocolo de exclusión mutua sea considerado como correcto, debe...

a) Únicamente estar libre de interbloqueo y evitar la inanición de procesos.
b) Únicamente garantizar el acceso atómico al recurso que protege.
c) Únicamente evitar condiciones de carrera entre procesos.
d) Ninguna de las otras respuestas es correcta.

50 Una aplicación concurrente...

a) Puede comunicarse con otros procesos del sistema a través de variables almacenadas en la zona de memoria compartida (variables static)
b) Siempre consta de al menos dos tareas en ejecución.
c) Será siempre más eficiente si emplea un pool para crear los hilos
d) Ninguna de las otras respuestas es correcta.

51 Una hebra tiene acceso...

a) A las variables declaradas en el resto de hilos de la aplicación.
b) A cualquier variable pública declarada como estática en la aplicación.
c) A todas las variables que se encuentran en el heap de la aplicación.
d) A las variables públicas declaradas en el resto de procesos en ejecución.

52 Considere el siguiente programa concurrente escrito en C++. ¿Cuál es su comportamiento en tiempo de ejecución?

```
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>

struct C {
        std::mutex lock;
        int xx = 0;

        void x() {std::lock_guard<std::mutex> guard(lock);
                  xx++;}
        void y() {std::lock_guard<std::mutex> guard(lock);
                  xx--;}
        void z() {lock.lock();
                  xx--;}
};
int main() {
        C cont;
        std::vector<std::thread> threads;
        for(int i=0; i<10; ++i) {
          threads.push_back(std::thread([&cont](){
            for(int j=0; j<100; ++j)
              cont.x();}));
        }
        for(int j=0; j<10; j++)threads[j].join();
          cont.y();
          cont.z();
        std::cout << cont.xx-10 << std::endl;
        cont.z();
        std::cout << cont.xx-10 << std::endl;
        return 0;
}
```

a) El programa imprime 999, 998 y termina normalmente.
b) El programa imprime 999, 989 y termina normalmente.
c) El programa imprime 988, 978 y queda bloqueado.
d) El programa imprime 988 y queda bloqueado.

53 Considere un escenario de concurrencia con Java donde se emplean estructuras de datos complejas con una alta carga de hebras concurrentes, del cuál se sabe, dada la naturaleza del problema, que el número de accesos concurrentes para la lectura de las hebras a las estructuras de datos es mucho mayor que el número de accesos para escritura. En estas condiciones, y para asegurar una solución al escenario que sea segura y optimice la vivacidad usted considera que la mejor solución es...

a) Proteger los accesos a las estructuras de datos mediante semáforos.
b) Proteger los accesos a las estructuras de datos mediante monitores
c) Proteger los accesos a las estructuras de datos mediante cerrojos de clase ReentrantLock.
d) Proteger los accesos a las estructuras de datos mediante transacciones Java/Clojure.

54 Suponga que desea desarrollar una aplicación de coeficiente de bloqueo nulo (C_b=0) en Python. Para maximizar la eficiencia de la aplicación, usted decide...

a) **Desarrollar una aplicación multiproceso que emplee el mecanismo de paso de mensajes para sincronizarlos y establecer una comunicación entre ellos.**
b) Desarrollar una aplicación multiproceso que emplee exclusivamente el modelo de memoria común para hebras de Python, a fin de sincronizarlos y establecer una comunicación entre ellos.
c) Desarrollar una aplicación multihilo que emplee variables compartidas para sincronizarlos y establecer una comunicación entre ellos.
d) Desarrollar una aplicación que emplee un pool para gestionar los hilos de la aplicación, y usar variables compartidas para sincronizar y establecer una comunicación entre las tareas enviadas al pool.

55 Se ha desarrollado con Java una versión paralela beta de un software destinado a simular interacciones intermoleculares entre moléculas de estructuras cristalinas tridimensionales regulares sobre una máquina multicore clasificada MIMD en la taxonomía de Flynn con acoplamiento fuerte y memoria común. Se plantea la explotación científica de la beta en un cluster de procesadores de alto rendimiento, aplicando la misma sobre estructuras cristalinas complejas, que exigen distribuir el cálculo entre los nodos del cluster y una alta tasa de comunicación internodal. En estas condiciones...

a) La beta es directamente ejecutable sobre el cluster.
b) **Es necesario reescribir la beta utilizando MPJ-Express para gestionar la alta tasa de transferencia de información entre tareas que sabemos que existe.**
c) Es necesario reescribir la beta utilizando control de acceso a la información utilizando memoria transaccional software con Java/Clojure.
d) Ninguna de las anteriores es correcta.

56 El método *notifyAll* de la API de Java...

a) Debe ser siempre ejecutado como última instrucción de los métodos públicos de cualquier monitor.
b) Debe ser siempre ejecutado como última instrucción de todos los métodos de cualquier monitor.
c) No garantiza que todas las hebras del wait-set asociado al objeto sobre el que se aplica reciben la señal.
d) **Ninguna de las otras respuestas es correcta.**

57 El empleo del mecanismo de comunicación entre procesos basado en el modelo de paso de mensajes es la opción más adecuada cuando...

a) La comunicación y/o sincronización entre los procesos se realiza en un sistema que emplea un esquema de memoria compartida..
b) La comunicación y/o sincronización entre los procesos se realiza en un sistema encuadrado dentro de la categoría SIMD de la taxonomía de Flynn.
c) **La comunicación y/o sincronización entre los procesos se realiza en un sistema que emplea un esquema de memoria distribuída.**
d) Se desea comunicar, pero no sincronizar, procesos distribuidos en un sistema encuadrado dentro de la categoría SIMD de la taxonomía de Flynn.

58 El uso de ejecutores de pool de threads en ambientes de programación paralela basados en multiprocesamiento simétrico tiene sentido, en términos de reducción de latencias de creación de hebras...

a) Siempre.
b) En problemas donde la carga de tareas paralelas es predecible, pequeña y constante.
c) Nunca.
d) En problemas donde la carga de tareas paralelas presenta variabilidad en el tiempo, inclusive con picos de alta demanda de tareas, sin un grado de predictibilidad claramente establecido.

59 Considere el siguiente programa concurrente escrito en Java. ¿Qué valor imprime como resultado de la ejecución?

```
public class cumbresTenebrosas extends Thread {
   public static int v = 0;
   public static int n = 1000;

   public cumbresTenebrosas() {this.start();}

   public void HeatCliff() {
      if (n>0) {
         n--;
         synchronized(this) {
            v--;
            this.HeatCliff();
         }
      }
   }

   public void run() {
      v++;
      this.HeatCliff();
      v--;
   }

   public static void main(String[] args) throws InterruptedException {
      cumbresTenebrosas ct = new cumbresTenebrosas();
      ct.join();
      System.out.println(v);
   }
}
```

a) Ninguno. El programa se bloquea.
b) 1000
c) 0
d) -1000

60 Se desea implementar una nueva clase en el lenguaje Java. Esta hereda de otra que implementa un método que está declarado en la superclase como synchronized. La subclase sobreescribe el método. ¿Cuál de las afirmaciones es correcta?

a) El mecanismo de herencia en el lenguaje Java no se aplica sobre el modificador synchronized, de tal forma que será necesario volver a escribir synchronized en el método sobreescrito para mantener este tipo de sincronización.
b) No será necesario controlar este método en exclusión mutua, ya que al ser un método con la cláusula synchronized en la superclase, este comportamiento también será heredado en las subclases.
c) El mecanismo de herencia en el lenguaje Java no permite heredar este tipo de métodos, por lo que ocurrirá un error de compilación.
d) Ninguna de las afirmaciones es correcta.

61 Los hilos creados por otro hilo...

a) No necesitan implementar un acceso en exclusión mutua para acceder a los recursos compartidos.
b) No necesitan implementar un acceso en exclusión mutua para acceder a los recursos compartidos, siempre y cuando el hilo padre acceda a dichos recursos en exclusión mutua.
c) Siempre deben acceder a los recursos compartidos en exclusión mutua.
d) Ninguna de las respuestas anteriores es correcta.

62 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 en el hilo hijo.
b) Pasando por referencia los hijos a las variables compartidas.
c) A través de un archivo.
d) Todas las respuestas son correctas.

63 Cada hilo comparte con su hilo padre...

a) El contador de programa.
b) El contador de programa y la pila de llamada.
c) La pila de llamadas y las variables estáticas.
d) Ninguna de las respuestas anteriores es correcta.

64 Un hilo creado por otro...

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 el mismo espacio de direcciones.
c) Debe esperar para adquirir el cerrojo de los métodos sincronizados del hilo padre.
d) Ninguna de las respuestas anteriores es correcta.

65 El entrelazado de instrucciones...

a) Sólo se produce entre hilos, pero no entre procesos.
b) Sólo se produce en arquitecturas multinúcleo.
c) Se produce en sistemas concurrentes y paralelos.
d) No se produce cuando se ejecutan instrucciones del sistema operativo.

66 ¿Cuál es el número máximo de clases que pueden heredar de la clase Thread en una aplicación escrita en Java?

a) No hay límite.
b) Sólo una clase puede heredar de Thread. El resto deben implementar la interfaz Runnable.
c) Sólo una, si se desea compartir información entre los hilos a través de variables estáticas.
d) Todas las clases deben heredar de la clase Thread.

67 El acceso en exclusión mutua a los recursos compartidos

a) Produce aplicaciones más eficientes.
b) Produce aplicaciones más eficientes cuando se utilizan primitivas de sincronización reentrantes.
c) Produce aplicaciones menos eficientes.
d) No afecta al rendimiento de las aplicaciones.

68 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 incrementará una unidad el número de veces que se puede realizar la operación H.wait(), sin que el hilo se quede bloqueado.
d) Ninguna de las respuestas anteriores es correcta.

69 Dadas dos variables A y B en Java, que apuntan a la misma dirección de memoria tal que A=B…

a) Cada una de ellas dispondrá de su propio Wait-Set asociado.
b) Cada una de ellas dispondrá de 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 sólo si ambas son de tipo Object.

70 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.

71 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 al método shutdown () seguido de un bucle que finalice cuando el método isTerminated () devuelve 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 a los hilos del pool de forma inmediata, sin esperar a la finalización de los que actualmente se encuentran ejecutándose.

72 Los monitores en C++…

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

73 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 de las respuestas anteriores es correcta.

74 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 encontrarse ejecutando instrucciones del protocolo de salida al mismo tiempo que otro hilo también las ejecuta.

75 Los protocolos de exclusión mutua basados en soluciones software…

a) Sólo funcionan correctamente si se accede en exclusión mutua a las variables compartidas.
b) Sólo funcionan correctamente si se accede por turnos a las variables compartidas.
c) Son más eficaces que los basados en soluciones hardware.
d) Ninguna de las respuestas anteriores es correcta.

76 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) Deben ser sincronizados si son públicos.

77 Un método sincronizado en Java…

a) Puede contener un bloque sincronizado que utilice la variable 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 dos primeras respuestas son correctas.

78 En el modelo de memoria de 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 anteriores respuestas son correctas.

79 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.

80 En el framework RMI, el servidor…

a) Puede efectuar la llamada a un método del cliente.
b) Hereda de la interfaz Runnable de Java.
c) Modela la clase AsyncEventHandler.
d) Ninguna de las respuestas anteriores es correcta.

81 Suponga que 500 hebras comparten el acceso a un objeto común cada una a través de su propia referencia, y ejecuta 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.

82 En Java, los cerrojos de la clase ReentrantLock se utilizan…

a) Para no tener regiones críticas 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.

83 Una hebra…

a) Puede estar detenida en un punto de su código por un bloqueo de exclusión mutua, mientras ejecuta código de 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 respuestas anteriores es correcta.

84 Usted debe desarrollar una aplicación bajo el framework RMI que controla un sistema de reserva de billetes de avión y sabe…

a) Que el sistema deberá ser concurrente.
b) Que habrá que 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 un DNS local a la máquina que ejecuta el servidor.

85 Considere un objeto en Java que dispone de tres métodos llamados m1, m2 y m3. El programador decide que deben ejecutarse en exclusión mutua, y para ello dispone de 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 release(). La sección crítica en todos los casos implica incrementar una variable del tipo int que inicialmente vale 3. Tres hebras externas, A, B y C, activadas dentro de una corutina ejecutan lo siguiente: *A.m1; B.m3; C.m2;*. Terminada la corutina, el programa imprime el valor de la variable, y este es…

a) 3, ya que las hebras se quedan bloqueadas y no lo modifican.
b) 6, pues las tres hebras hacen su incremento de forma segura
c) 5, puesto que 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.

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

87 Es conocido que la anidación que 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) Anidar transacciones.
c) Verificar de manera formal las propiedades de corrección de nuestro código.
d) Utilizar el API de alto nivel del lenguaje.

88 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) No puede ejecutar código paralelo y actuar como coprocesador de la CPU.
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.

89 El uso de ejecutores en lenguaje Java supone…

a) Optimizar siempre el rendimiento de la aplicación.
b) Procesar indistintamente tareas Callable o Runnable a partir 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 datos compartidos.

90 El uso de variables volatile en el ámbito de la concurrencia…

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

91 El resultado de paralelizar una aplicación sobre una máquina de ocho cores ofrece un speedup de 2.75. Eso significa que…

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

92 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.

93 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 las variables de condición si son necesarias.
d) Ninguna de las anteriores es correcta.

94 La implementación del concepto teórico de región crítica 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 correctas.

95 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 correctas.

96 Se desea paralelizar un algoritmo para el que se ha determinado que C_b=0, y tiene que trabajar sobre una máquina de 8 cores lógicos para procesar un vector de 10^6 componentes sin interdependencia entre ellas. Con estos datos, usted decide:

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

97 La reentrancia para código synchronized en el lenguaje Java…

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

98 Usted debe determinar el valor de Cb óptimo para una aplicación con paralelismos a nivel de hebras que acaba de escribir. Para ello…

a) 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) Busca el mínimo de la curva Tiempo = f (Cb) mediante experimentación, lo determina de forma aproximada y deduce a partir de aquí cuantas hebras necesita tener en su aplicación.
d) No es posible determinar esto de forma adecuada sin un benchmark profesional. Puesto que su código tiene latencias de E/S y de red, su experiencia como programador le dice que Cb es aproximado a 0,5.

99 Los monitores escritos en lenguaje C#…

a) Permiten varias colas de espera por condición.
b) No permiten colas de espera por condición; funcionan exclusivamente para proteger recursos compartidos bajo control de exclusión mutua.
c) Funcionan igual que los monitores en Java con el API de alto nivel
d) Ninguna de las anteriores es cierta.

100 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 máquina.
d) Sólo hay realmente un hilo daemon correspondiente al recolector de basura.

101 La concurrencia basada en un esquema de paso de mensajes es adecuada…

a) Cuando se tienen tareas concurrentes que comparten memoria.
b) Cuando hay exclusión mutua entre datos compartidos por diferentes tareas concurrentes.
c) Cuando se tiene paralelismo de datos por división del dominio de datos en memoria común entre las tareas.
d) Ningún caso de los anteriores es correcto.

102 La especificación de tiempo real para Java (JRTS) define un modelo de planificación de 28 prioridades y carácter expulsivo, lo cuál supone…

a) Que el programador es completamente responsable de analizar e implementar la planificabilidad de las soluciones de tiempo real que programa.
b) Que el programador debe, siempre, implementar una planificación mediante ejecutivo cíclico.
c) Que no es necesario diseñar una planificación, ya que la expulsividad del modelo lo hace innecesario.
d) Que el programador debe solventar la planificación recurriendo a objetos soportados mediante Schedulable.

103 La presencia del GIL (Global Interpreter Lock) en el lenguaje Python supone cuando programamos con este lenguaje y el módulo threading...

a) Presencia de concurrencia pero no de paralelismo.
b) Ausencia de concurrencia y de paralelismo, ya que el GIL impide ambas.
c) Presencia de paralelismo pero no de concurrencia.
d) Presencia de paralelismo MIMD según la taxonomía de Flynn.

104 Considere un cluster de procesadores del cuál se sabe que la red de comunicaciones entre nodos es muy lenta para los estándares actuales. Cada nodo del cluster dispone de 16 núcleos de ejecución con 128GB de memoria RAM. Se desea ejecutar en el cluster la simulación de un modelo de plegamiento de proteínas utilizando un esquema de paso de mensajes, y se sabe que el algoritmo base de la simulación implica una alta tasa de comunicación entre tareas paralelas. En estas condiciones:

a) Usted se empeña en intentar optimizar más el algoritmo.
b) Se conforma con lo que hay, supone que $C_b = 0$ y ejecuta la simulación.
c) Encarga a su equipo de programadores una labor de benchmarking e intenta determinar C_b de forma más precisa.
d) Nada de lo anterior tiene sentido.

105 Suponga un conjunto de hebras en Java que realizan de forma concurrente una operación de autoincremento sobre la variable compartida x. Dicha variable x es declarada volatile, por tanto:

a) Las operaciones de lectura y escritura en memoria y la de auto incremento son atómicas en su conjunto, por lo que el resultado será una operación de hebra segura.
b) Las operaciones de lectura y escritura en memoria no son atómicas, pero sí la de autoincremento, por lo que el resultado será una operación atómica realizada entre dos operaciones no sincrónicas y por tanto es una operación de hebra no segura.
c) Ni las operaciones de lectura y escritura en memoria, ni la de auto incremento son atómicas, por lo que el resultado será una operación de hebra no segura.
d) Las operaciones de lectura y escritura en memoria son atómicas, pero no la de autoincremento, por lo que el resultado será una operación de hebra no segura realizada entre dos operaciones sincrónicas.

106 La programación en Java con ejecutores tiene sentido...

a) Cuando se implanta un servidor de red con tareas soportadas por la interfaz Callable o Runnable.
b) Cuando se tienen muy pocas tareas soportadas mediante expresiones lambda.
c) Cuando se implanta un servidor de red con el framework RMI.
d) Cuando se implanta una convolución matricial sobre una arquitectura MIMD de 4 cores lógicos.

107 Usted ha determinado que una solución concurrente lo más eficiente posible a un determinado problema requiere el uso de dos conjuntos de datos disjuntos entre sí. A partir de aquí, usted concluye...

a) Que necesita utilizar un monitor único que engloba a ambos conjuntos bajo exclusión mutua.
b) Que necesita utilizar dos monitores que engloban bajo exclusión mutua a cada uno de los conjuntos de datos.
c) Que necesita proteger ambos conjuntos de datos bajo un semáforo S que cuyo valor inicial es de 1.
d) Que necesita utilizar una única región crítica condicional que engloba a ambos conjuntos bajo exclusión mutua.

108 Una de las principales diferencias entre un semáforo y una variable de condición es:

a) Que en ambos casos se indiza una cola de procesos bloqueados.
b) La variable de condición carece de estado, por lo que la instrucción send(C) no tiene efecto si la cola está vacía.
c) La invocación de la primitiva wait(S) en un semáforo S siempre bloquea al hilo.
d) La gestión de procesos bloqueados en un semáforo S siempre es FIFO.

109 Los métodos wait() y notify() de la clase Condition en Python:

a) Son atómicos.
b) Requieren el uso de condiciones de guarda al utilizar wait.
c) Utilizan una semántica de señalización SC.
d) Todas las respuestas anteriores son correctas.

110 Cuando implementamos un monitor utilizando C++:

a) Podemos emplear tantas variables de condición como sea necesario.
b) No es necesario llamar al método wait habiendo tomado previamente el bloqueo sobre un cerrojo.
c) Siempre tenemos que utilizar condiciones de guarda.
d) Sabemos que utiliza un único wait -set.

111 Durante la programación concurrente con JAVA, la reentrancia es una característica...

a) Disponible únicamente en el API estándar con cerrojos de clase ReentrantLock.
b) Disponible únicamente en el API estándar con métodos synchronized.
c) Disponible únicamente en el API estándar con métodos synchronized cuando el método es recursivo.
d) Disponible tanto en el API estándar como en el API de alto nivel.

112 En un escenario de programación concurrente bajo memoria transaccional software con datos compartidos entre varias tareas concurrentes y protegidos mediante transacciones:

a) Es posible una única operación de commit sin operaciones de rollback cuando hay múltiples tareas escritoras.
b) Es posible una única operación de rollback y múltiples operaciones de commit cuando hay igual número de tareas lectoras y escritoras superior a diez para cada tipo de tarea.
c) No se requieren operaciones de commit ni de rollback cuando todas las tareas son escritoras.
d) No se requieren operaciones de commit ni de roll -back cuando todas las tareas son lectoras.

113 La planificación mediante un ejecutivo cíclico en un sistema de tiempo real tiene sentido...

a) Cuando el sistema de tiempo real además tiene carácter de sistema empotrado.
b) Cuando una de las tareas del sistema es periódica, y se sabe que se activa cada T segundos de tiempo del sistema.
c) Cuando existe un orden total iterado entre todas las tareas del sistema, conocido a priori.
d) Cuando existe otro orden parcial entre todas las tareas del sistema, conocido a priori.

114 Utilizando MPJ -Express es posible tener una cita o “rendezvouz” entre dos tareas comunicadas...

a) Utilizando Bcast/Reduce.
b) Utilizando Ssend/Receive.
c) Utilizando Gather/Scatter.
d) No podemos tener una cita entre dos tareas.

115 Una solución cliente-servidor con sockets en Java utiliza un servidor multihebrado mediante un pool de hebras que involucra a estructuras de datos compartidas que son listas numéricas. Diferentes clientes pueden trabajar sobre la misma o sobre diferentes listas. Usted desea un servidor lo más eficiente posible y para ello:

a) Modela sus listas mediante clases contenedoras autosincronizadas.
b) Engloba el acceso a todas las listas bajo el control de un objeto de clase Semaphore
c) Encapsula a todas las listas en un único monitor utilizando el API de alto nivel para concurrencia.
d) Engloba el acceso a todas las listas bajo el control de un objeto de clase ReentrantLock.

116 En una solución basada en el framework RMI el único objeto que soporta el servicio en el lado del servidor gestiona diversos contadores numéricos que almacenan información sobre los clientes que solicitan el servicio, y que deben gestionarse de la forma más eficiente posible. Dado lo anterior, usted:

a) Sabe que no necesita control de exclusión mutua, dado que el servidor creará nuevos objetos servidores de forma automática que lo hacen innecesario.
b) Sabe que no necesita control de exclusión mutua, puesto que no programó hebras en su servidor cuando escribe código RMI.
c) Controla la exclusión mutua sobre todos los contadores con objetos de clase Semaphore.
d) Controla la exclusión mutua con objetos atómicos.

117 Se desea efectuar una convolución tridimensional utilizando un kernel de convolución de 3 x 3 x 3 celdas sobre una matriz también tridimensional de 10^9 x 10^9 x 10^9 celdas mediante procesamiento paralelo utilizando el lenguaje C++. Su plataforma hardware es un sistema empotrado con una memoria que no permite alojar más que le matriz de datos, el kernel de convolución, y el código objeto ejecutable de su programa (además del sistema operativo), si bien el disco duro tiene tanto espacio libre como necesite, aunque presenta tiempos de lectura-escritura que pueden considerarse lentos. Con los datos anteriores, usted:

a) Decide que no es posible solucionar el problema con esos presupuestos.
b) Utiliza dos copias de la matriz, una en memoria principal para escritura, otra en el disco duro para lectura implementa una solución paralela con C_b = 0 y divide la matriz original entre las tareas paralelas.
c) Utiliza dos copias de la matriz, una en memoria principal para escritura, otra en el disco duro para lectura implementa una solución donde determina cuánto vale C_b y divide la matriz original entre las tareas paralelas
d) Programa una solución secuencial, que es la única viable con las especificaciones proporcionadas.

118 Los algoritmos de control de exclusión mutua con memoria común...

a) Generan sobrecargas de ejecución.
b) Son altamente transportables.
c) No dependen del modelo de memoria del lenguaje en que se implementan.
d) Implican el uso de colas de espera ocupada para gestionar la exclusión mutua.

119 Se han desarrollado dos soluciones en Python que multiplican matrices paralelamente utilizando los módulos threading y multiprocessing, y obtenido los speedups con ambas soluciones para una arquitectura de 8 cores lógicos, suponiendo que $C_b = 0$. Estos speedups pueden ser:

a) Igual a 0,99 para threading y 7,14 para multiprocessing.
b) Iguales a 1,00 para threading y para multiprocessing, ya que el GIL impide un paralelismo real en Python con ambos módulos.
c) Igual a 5,38 para threading y 7,14 para multiprocessing.
d) Igual a 7,14 para threading y 1,0 multiprocessing.

120 Es un sistema de tiempo real, las tareas que se ejecutan en la región de memoria inmortal...

a) Pueden ser expulsadas por el gc.
b) Pueden ser expulsadas por tareas de menor prioridad.
c) En la memoria inmortal las tareas no pueden ser expulsadas.
d) Pueden ser expulsadas.

121 Dados los siguientes procesos concurrentes:

```
N: Integer := 0
M: Integer := 1

Task Body P1 is          Task Body P2 is
begin                    begin
  N := N + 1;              N := 10;
  M := M + 1;              M := 10;
end P1;                  end P1;
```

Seleccione la afirmación correcta según las Condiciones de Berstein.

a) Los procesos no pueden ejecutarse concurrentemente de forma segura ya que la intersección de algunos de los conjuntos de escritura no es vacía.
b) Los procesos pueden ejecutarse concurrentemente de forma segura ya que los conjuntos de lectura de las instrucciones del proceso P2 son vacíos.
c) El conjunto de lectura de la instrucción N := N + 1; contiene los elementos {N, 1}.
d) En la memoria inmortal las tareas no pueden ser expulsadas.

122 En la especificación JRTS de tiempo real para el lenguaje Java, el algoritmo que implementa el *garbage collector*...

a) Tiene acceso a todos los tipos de memoria que la especificación define.
b) puede interrumpir sumariamente la ejecución de cualquier objeto *schedulable* activo en el sistema.
c) Tiene pleno acceso a la región de memoria *InmortalMemory*.
d) Nada de lo anterior es cierto

123 El método *tryAcquire()*, en su versión no parametrizada...

a) Decrementa siempre en una unidad el valor del semáforo si este tiene un valor mayor que cero.
b) Nunca produce el bloqueo del hilo que lo invoca.
c) Devuelve false si el valor del semáforo es 0.
d) Todas las respuestas son correctas.

124 ¿Qué puede afirmarse sobre el paquete *java.util.concurrent.atomic*?

a) Java no soporta la programación concurrente de forma segura a variables tratadas a través de forma atómica.
b) Es un conjunto de clases que gestiona el acceso concurrente a todos los tipos de variables no primitivas, sea cual sea su clase, de forma segura.
c) Aunque un tipo tenga su versión atómica, esto no quiere decir que todos sus métodos soporten concurrencia segura.
d) Ninguna de las otras opciones es correcta.

125 En una aplicación multihilo desarrollada en el lenguaje Java.

a) Los hilos de la aplicación pueden leer pero no modificar las variables de la aplicación a las que tengan acceso por su nivel de protección (public, protected o private).
b) Todos los hilos de la aplicación pueden modificar todas las variables de la aplicación a las que tengan acceso por su nivel de protección (public, protected o private).
c) Todos los hilos de la aplicación pueden modificar todas las variables de la aplicación, independientemente de su nivel de protección (public, protected o private).
d) Ninguna de las otras respuestas es correcta.

126 Una aplicación crea tres hilos (primer nivel) que a su vez crean otros cuatro hilos (segundo nivel) cada uno. Seleccione la afirmación correcta.

a) Los hilos del segundo nivel tienen acceso únicamente a las variables declaradas en su hilo padre, pero no a las variables declaradas en el resto de hilos del primer nivel.
b) Los hilos del segundo nivel tienen acceso a las variables declaradas en el hilo principal y en los hilos de primer nivel, pero no en el resto de hilos del segundo nivel.
c) Los hilos del primer nivel tienen acceso a las variables declaradas en el hilo principal y en sus hilos hijos del segundo nivel, pero no en el resto de hilos del segundo nivel.
d) El acceso de los hilos a las variables declaradas en la aplicación dependerá de su nivel de protección (public, protected, private).

127 En un sistema concurrente los datos compartidos frente a accesos simultáneos de las hebras están protegidos bajo control de memoria transaccional software, con lo cual:

a) La ausencia de deadlocks está garantizada.
b) Podemos tener múltiples hebras efectuando lecturas concurrentes sobre los datos compartidos.
c) Las hebras que efectúan escrituras concurrentes sobre los datos compartidos pueden necesitar efectuar roll-backs para lograr una transacción de escritura completada con éxito (commit).
d) Todas las anteriores son ciertas.

128 Considere el siguiente código multihebrado escrito en Python. El resultado que ofrece su ejecución cuando la variable iter se carga con un valor igual a 1000 es:

```
from threading import Thread
from threading import Lock
import time

shared_cont = 0
lock        = Lock()

def myThread(iter):
    global shared_cont
    for i in range(iter):
        shared_cont+=1
        lock.acquire()

if      __name__ == '__main__':
    iter     = int(input('iteraciones?'))
    myThread1 = Thread(target=myThread, args=(iter,))
    myThread2 = Thread(target=myThread, args=(iter,))
    start=time.time()
    myThread1.start()
    myThread2.start()
    myThread1.join()
    myThread2.join()
    lock.release()
    end=time.time()-start
    print('valor: ',shared_cont)
    print('tiempo: ',end)
```

a) 2000
b) Un número entre 0 y 2000
c) No hay resultados; el código permanece en bloqueo
d) 4000

129 Se desea proteger un conjunto de datos bajo control de exclusión mutua; se sabe que el acceso de las hebras a los mismos va además a requerir de necesidades de sincronización. Se diseña un monitor en C++ que proporcione soporte a las necesidades identificadas. En estas condiciones:

a) Debemos saber que dicho monitor va a permitir una única cola de espera por condición, con señalización SX.
b) Sabemos que dicho monitor funciona igual que uno escrito en lenguaje Java con el API estándar de control de la concurrencia.
c) Sabemos que dicho monitor permite implementar múltiples colas de espera por condición.
d) Sabemos que un monitor implementado en C++ únicamente permite resolver las necesidades de control de la exclusión mutua.

130 De las siguientes circunstancias, ¿cuáles pueden inducir un interbloqueo (deadlock)?

a) El acceso a una sección crítica en exclusión mutua.
b) La espera circular.
c) Una mala definición de un protocolo de exclusión mutua.
d) Cualquiera de ellas.

131 Considere el siguiente código en C++. El resultado que ofrece su ejecución es:

```
#include <iostream>
#include <mutex>

using namespace std;

struct Contador{
        mutex mut;
        int val=0;

        void inc(int a){
                lock_guard<mutex> cerrojo(mut);
                val+=a;
        }

        void dec(int a){
                lock_guard<mutex> cerrojo(mut);
                val-=a;
        }

        void todas(int a, int b){
                lock_guard<mutex> cerrojo(mut);
                inc(a); dec(b);
        }
};

int main()
{
  Contador contador;
  contador.todas(100,50);

  cout << contador.val;

  return 0;
}
```

a) 150.
b) 50.
c) -50.
d) Se produce un bloqueo.

132 Al equipo de ingenieros que usted dirige se le encarga la implementación de un sistema de venta de entrada on-line; se toma la decisión de soportar el sistema mediante una arquitectura distribuida utilizando el framework RMI. A partir de este punto, usted sabe que...

a) Necesita programar explícitamente el multihebrado que el lado del servidor requiere.
b) Que puede distribuir servidor y DNS entre diferentes máquinas.
c) Que el uso de generación dinámica de resguardos le permite prescindir de un DNS local a la máquina donde el servidor se ejecuta.
d) **Que necesite utilizar control de exclusión mutua explícito sobre los datos compartidos en el lado del servidor.**

133 Una aplicación en Java crea dos hilos que a su vez crean otros cinco hilos cada uno. Cuando esa aplicación es ejecutada...

a) Se crean en el sistema dos procesos y cinco hilos.
b) Se crean en el sistema 10 procesos.
c) Se crea en el sistema un único proceso.
d) Se crean en el sistema el mismo número de hilos que de procesos.

134 La técnica de programación paralela conocida como multiprocesamiento simétrico, que utiliza paralelismo mediante división del dominio de datos entre las tareas

a) Está soportada por Java y C++.
b) Puede implementarse con lenguajes no orientados a objeto, como C.
c) Es adecuada para resolver con paralelismo problemas con nubes de datos reticulares muy estructuradas.
d) Todas las anteriores son ciertas.

135 Cuando usamos un cerrojo de clase ReentrantLock en Java...

a) Podemos definir cuantos hilos accederán al mismo tiempo al recurso compartido.
b) No será posible hacer uso de recursos más avanzados como los objetos de la clase Condition.
c) Es importante envolver el código en un bloque try/finally para asegurar el desbloqueo en caso de que se produzcan excepciones.
d) Todos los recursos que utilicemos en los métodos con la palabra clave ReentrantLock estarán protegidos ante la exclusión mutua.

136 ¿Cuál de las siguientes afirmaciones sobre el concepto de paso de mensajes asíncrono es correcta?

a) Normalmente, con el uso de buzones, los procesos que se bloqueen primero serán los primeros en desbloquearse.
b) Dicho concepto se apoya en buzones, que pueden estar soportados por colas de mensajes, pipes, etc.
c) Con carácter general, el emisor puede enviar tantos mensajes como desee, con el límite del tamaño del buzón.
d) Todas las respuestas son correctas.

137 Considere una solución multihebrada al problema de convolución de una matriz de alta dimensión. En estas condiciones:

a) La solución multihebrada utiliza paralelismo funcional.
b) La solución multihebrada acelera la ejecución siempre si se utiliza Python.
c) La solución multihebrada utiliza paralelismo de datos.
d) Ninguna de las otras respuestas es correcta.

138 Considere un objeto de clase CopyOnWriteArrayList que contiene referencias a objetos Coche, definida por el usuario (aclaración: CopyOnWriteArrayList<Coche>).

a) Las operaciones de inserción, actualización y borrado de los elementos (referencias) en el objeto de clase CopyOnWriteArrayList se realizan de forma segura.
b) Las operaciones de inserción, actualización y borrado de los elementos (referencias) en el objeto de clase CopyOnWriteArrayList, así como la modificación de los atributos de los objetos de tipo Coche, se realizan de forma segura.
c) Solo se garantiza que la operación de inserción de elementos (referencias) en el objeto de clase CopyOnWriteArrayList se realiza de forma segura, pero no la de eliminación ni actualización.
d) El objeto de clase CopyOnWriteArrayList garantiza un acceso más rápido a los elementos del array cuando se trata de operaciones de escritura.

139 ¿Qué ocurre si en una aplicación desarrollada empleando el esquema habitual de sincronización basada en cerrojos (ReentrantLock) y variables de condición (Condition) se aplica el método wait() a una instancia de la clase Condition en lugar del método habitual await()?

a) La aplicación se comportará exactamente igual siempre que la variable condición esté asociada correctamente al cerrojo.
b) La invocación del método wait() provocará un error, puesto que ha de emplearse siempre dentro de un bloque o método de tipo synchronized.
c) La invocación del método wait() no tendrá ningún efecto sobre los hilos que lo invoquen.
d) La aplicación se comportará exactamente igual, puesto que el método await() es una especialización (sobreescribe) al método wait().

140 Cuando un hilo llama a una operación de un monitor implementado en Java con el API estándar, pero ya hay otro hilo ejecutando la misma operación, ¿qué ocurre?

a) Se produce un interbloqueo.
b) El hilo que llama a la operación se quedará en espera, hasta que el hilo que actualmente ejecuta la operación termine.
c) Comenzará la ejecución de la operación con normalidad, solo que se gestionará el acceso a los recursos compartidos a través de los métodos wait y signal.
d) El hilo que llama a la operación es enviado al wait-set, donde quedará en espera y continuará su ejecución una vez que el otro hilo termine de usar el monitor.

141 Para procesar de forma paralela y eficiente en Python una nube de datos reticulada y multidimensional utilizando como base del paralelismo el multithreading que el lenguaje ofrece, debemos...

a) Aplicar cuidadosamente la ecuación de Subramanian para determinar el número de hebras necesarias
b) Transferir a las hebras mediante el constructor el subsegmento de datos que deben procesar.
c) Considerar cuidadosamente las necesidades de sincronización inter-hebras que el problema pueda requerir
d) No perder el tiempo paralelizando; con las condiciones dadas, no podemos tener un paralelismo real.

142 Considere el siguiente programa multihebrado en Python. Indique la salida impresa que produce.

```
import threading
x = 0

def increment_global():
   global x
   x += 1

def taskofThread(lock):
   for _ in range(1000000):
      lock.acquire()
      increment_global()
      lock.release()

def taskofThreadToo():
   for _ in range(2000000):
      increment_global()

def main():
   global x
   x = 0
   lock = threading.Lock()
   t1 = threading.Thread(target = taskofThread, args = (lock,))
   t2 = threading.Thread(target = taskofThread, args = (lock,))
   t3 = threading.Thread(target = taskofThreadToo, args = ())
   t1.start()
   t2.start()
   t3.start()
   t1.join()
   t2.join()
   t3.join()

if __name__ == "__main__":
      main()
      print(x)
```

a) No hay salida impresa; el problema se bloquea.
b) Un dígito que puede estar entre 3000000 y 4000000.
c) Un dígito que puede estar entre 0 y 2000000
d) Un dígito igual a 0.

143 Sea H_U el número de hilos del espacio de usuario creados por una aplicación, y H_S el número de hilos del sistema que finalmente asigna el sistema operativo para la ejecución de una instancia de esa aplicación. En esta situación, se puede afirmar que...

a) $H_U >= H_S$
b) $H_U < H_S$
c) $H_U = H_S$
d) $H_U > H_S$

144 Un bloque *dosync* en Clojure...

a) Permite envolver el código dentro de un ámbito con control transaccional, donde se accede a los datos compartidos, permitiendo el acceso seguro.
b) Funciona exactamente igual que los bloques synchronized en Java
c) Se utiliza para crear y modificar entidades mutables
d) Todas las opciones son correctas.

145 Considere el siguiente programa multihebrado en Java. Indique la salida impresa que produce:

```
public class Deadlock {
  static Object region_A = new Object();

  public static void main(String[] args){
   final Object region_B = new Object();

   Thread Hilo_A = new Thread(new Runnable(){
     public void run(){
           synchronized(region_A){
             synchronized(region_B){
                   System.out.println("hilo A");
             }
           }
     }
   });

   Thread Hilo_B = new Thread(new Runnable(){
         public void run(){
           synchronized(region_B){
             synchronized(region_A){
           System.out.println("hilo B");
                 }
           }
     }
   });

   Hilo_B.start();
   Hilo_A.start();
 }
}
```

a) La aplicación quedará siempre bloqueada sin llegar a imprimir ningún mensaje
b) Primero se imprimirá el mensaje "hilo A" y a continuación la aplicación quedará bloqueada.
c) Primero se imprimirá el mensaje "hilo A", y a continuación el mensaje "hilo B".
d) La aplicación puede imprimir los mensajes "hilo A" e "hilo B" o quedar bloqueada sin llegar a imprimir ningún mensaje.

146 ¿Cuál de las siguientes afirmaciones es correcta?

a) Un proceso posee un contador de programa, que podrá ser compartido por múltiples hebras de cara a gestionar el acceso ordenado a los datos.
b) En un proceso, múltiples hebras tendrán acceso a la misma pila de llamadas, aunque opcionalmente pueden tener una pila independiente.
c) Las hebras de un proceso comparten los recursos del sistema, por lo que la ejecución de múltiples hebras puede afectar al rendimiento global del programa.
d) Todas las afirmaciones son correctas.

147 ¿El empleo de una solución paralela supone siempre una mejora de rendimiento con respecto de una solución secuencial que resuelve el mismo problema?

a) Dependerá de si se trata de un sistema con memoria compartida o distribuida.
b) Sí, en todos los casos.
c) Dependerá del número de cores de la arquitectura donde se ejecute la solución paralela, del problema a resolver y de la calidad de la solución paralela.
d) Dependerá únicamente del número de cores de la arquitectura donde se ejecute la solución paralela.

148 Desea diseñar un programa en Java utilizando RMI. ¿Para qué sirve la interfaz que constituye el elemento nuclear de una aplicación RMI?

a) Para lanzar el servidor, previo registro en un DNS.
b) Para establecer el protocolo de comunicaciones cliente-servidor.
c) Para registrar un objeto remoto en el servidor, de cara a invocar los métodos que implementa.
d) La interfaz es una parte opcional y solo sirve para que el cliente conozca más rápido los métodos disponibles en el servidor.

149 Considere el siguiente programa multihebrado escrito en C++. Indique la salida impresa por pantalla que produce.

```
#include <iostream>
#include <thread>
#include <vector>
using namespace std;


int main()
{
        vector<thread> hilos;
        int num = 1000;
        int contador = 0;

        for(int i = 0; i<num; i++)
                hilos.push_back(thread([&](){for(int i = 0; i<num; i++)contador++;}));

        for(auto& thread : hilos){thread.join();}

        cout << contador << endl;

        return 0;
}
```

a) 0
b) 1000000
c) Un número entre 100000 y 1000000
d) Ninguna de las anteriores es correcta.

150 Considere el siguiente programa multihebrado escrito utilizando las capacidades de multithreading ofrecidas por el lenguaje Clojure, que crea tres hebras diferentes cuyos motores de ejecución están constituidos por la sentencias de impresión println, en el ámbito de las respectivas funciones fn []. Las hebras se activan a efectos de ser planificadas para ejecución con las llamadas a .start. Las llamadas a .join se comportan igual que en Java. Indique qué salida impresa produce el programa.

```
(def threada (Thread. (fn [] (println 1 2 3 4 5 6))))
(.start threada)
(.join  threada)
(def threadb (Thread. (fn [] (dosync(println 1 2 3 4)))))
(def threadc (Thread. (fn [] (dosync(println 1 2 3 4)))))
(.start threadb)
(.start threadc)
```

a) Diferentes entrelazados de las distintas sentencias de impresión
b) 1 2 3 4 5 6
1 2 3 4
1 2 3 4
c) 1 2 3 4
1 2 3 4 5 6
1 2 3 4
d) 1 2 3 4
1 2 3 4
1 2 3 4 5 6

151 Se desea desarrollar una aplicación en Java que, por su naturaleza, crea hilos de forma masiva. ¿Es conveniente utilizar de forma exclusiva la clase Thread para la gestión de los hilos?

a) No, en estos casos es mejor utilizar un Pool de Threads, ya que solo se creará una vez y nos permite reutilizar los hilos de forma automática.
b) No, ya que la clase Thread limita el número de hilos con los que se puede trabajar.
c) Sí, ya que la clase Thread provee de un modelo de planificación por prioridades seguro y eficaz, y que es respetado íntegramente por el planificador del sistema operativo.
d) Sí, siempre que conozcamos el número de hilos, para así declararlos y gestionar sus prioridades con seguridad.

152 Una barrera implementada con la clase CyclicBarrier en Java...

a) Sirve como punto de espera para la sincronización de los hilos, que quedarán en espera hasta que todos los que se esperan lleguen a ella.
b) Es útil cuando en la siguiente fase de ejecución se necesita que todos los hilos comiencen de forma simultánea.
c) Permite sincronizar un número determinado de hilos de cara a unificar los resultados parciales obtenidos por los mismos.
d) Todas las opciones son correctas.

153 Las soluciones que la programación concurrente ofrece para resolver problemas de sincronización son aplicables también a la programación paralela...

a) Independientemente de si se trata de sistemas con memoria común o distribuida.
b) Pero solo cuando se trata de sistemas paralelos con memoria común.
c) Y las soluciones que la programación paralela ofrece para resolver problemas de sincronización son aplicables también a la programación concurrente.
d) Ninguna de las otras afirmaciones es correcta.

154 Suponga que usted desarrolla una aplicación en la cual todos los posibles pares de instrucciones cumplen las condiciones de Berstein. En ese caso...

a) No es posible desarrollar una solución paralela de la aplicación pero sí concurrente.
b) Es necesario emplear primitivas de sincronización para efectuar una ejecución paralela segura de la aplicación.
c) Es necesario emplear primitivas de sincronización para efectuar una ejecución concurrente segura de la aplicación.
d) El máximo Speedup que podrá alcanzar la aplicación coincidirá con aproximadamente el número de núcleos de la arquitectura donde se ejecute la aplicación.

155 Suponga que está usted desarrollando una aplicación distribuida empleando el framework RMI cuyo propósito es efectuar computación numérica con paralelismo de datos sobre una nube de datos estructurada procedente del cliente. En la aplicación se cuenta con un único cliente y un único objeto servidor corriendo en un una arquitectura de ocho cores. ¿Es posible obtener una mejora de rendimiento si el programador crea varios hilos para resolver las peticiones que le llegan por parte del cliente?

a) Solo se obtendrá una mejora de rendimiento si el programador emplea un pool de threads para resolver cada una de las peticiones del cliente.
b) No se obtendrá nunca una mejora de rendimiento ya que, de por sí, cada petición por parte de los clientes supone la creación de un hilo para ser atendida.
c) En el lado del servidor ya se emplea un multihebrado implícito para atender cada una de las peticiones de los clientes, por lo que no es posible crear más hilos de forma explícita.
d) Sí, sería posible.

156 Considere un monitor Java que encapsula como recurso protegido un array de referencias a objetos de clase Libro. Dicha clase modela el concepto de libro, e incluye atributos como autor o título, además métodos observadores y modificadores estándar... Los programadores del sistema trabajan en dos grupos diferentes que han consensuado el uso del array de objetos, y uno de los grupos (sin conocimientos de programación concurrente) desconoce que el otro ha tomado la decisión de encapsular el array en un monitor, y utiliza dicho array tal cual. En estas condiciones…

a) No podemos asegurar que los objetos referenciados desde el array sean totalmente seguros frente a hebras concurrentes del sistema.
b) El array de objetos estará controlado de forma segura frente a concurrencia siempre que el monitor haya sido escrito con señalización SX.
c) El array de objetos estará controlado de forma segura frente a concurrencia siempre que el monitor haya sido escrito utilizando una combinación de métodos synchronized y cerrojos de clase ReentrantLock.
d) El array de objetos estará controlado de forma segura frente a concurrencia siempre que el monitor haya sido escrito con el API de alto nivel y señalización SC.

157 Considere el siguiente programa multihebrado escrito en C++. Indique la salida impresa.

```
#include <iostream>
#include <thread>
#include <mutex>
#include <condition_variable>
using namespace std;

struct Buffer{
  int* buffer;
  int  tam, In_Ptr, Out_Ptr, cont;
  mutex em;
  condition_variable not_full, not_empty;

  Buffer(int capacidad){
    tam=capacidad; In_Ptr=0; Out_Ptr=0; cont=0; buffer = new int[tam];
  }

  ~Buffer(){delete[] buffer;}

  void insertar(int dato){
    unique_lock<mutex> l(em);
    while(cont==tam){not_full.wait(l);}
    buffer[In_Ptr]=dato;
    In_Ptr=(In_Ptr+1)%tam;
    ++cont;
    not_full.notify_one();
    l.unlock();
  }

  int extraer(){
    unique_lock<mutex> l(em);
    while(cont==0){not_empty.wait(l);}
    int result = buffer[Out_Ptr];
    Out_Ptr=(Out_Ptr+1)%tam;
    --cont;
    not_full.notify_one();
    return(result);
    l.unlock();
  }
};

 void productor(int id, Buffer& buffer){
    for(int i=0; i<100; ++i){
      buffer.insertar(i);
      cout<<"Productor "<<id<<" inserto "<<i<<endl;
    }
  }

  void consumidor(int id, Buffer& buffer){
   for(int i=0; i<50; ++i){
    int valor = buffer.extraer();
    cout<<"Consumidor "<<id<<" extrajo "<<valor<<endl;
   }
  }

int main(){
  Buffer buffer(1);
  thread c1(consumidor, 0, ref(buffer));
  thread c2(consumidor, 1, ref(buffer));
  std::this_thread::sleep_for (std::chrono::seconds(10));
  thread p1(productor, 0, ref(buffer));
  c1.join(); c2.join(); p1.join();
  return(0);
}
```

a) El programa imprime "Productor 0 inserto 0" y se bloquea.

b) El programa imprime "Productor 0 inserto 0", "Consumidor 0 extrajo 0" y termina normalmente.

c) El programa imprime "Consumidor 0 extrajo 0", "Productor 0 inserto 0", y termina normalmente.

d) El programa imprime "Productor 0 inserto 0", "Consumidor 1 extrajo 0" y termina normalmente.

158 Considere el siguiente programa multihebrado en Java. Indique la salida impresa que produce.

```
import java.util.concurrent.*;
import java.util.concurrent.locks.*;

public class OtroDeadlock {
  static ReentrantLock cerrojo = new ReentrantLock();

  public static void main(String[] args){
   final Object region_B = new Object();

   Thread Hilo_A = new Thread(new Runnable(){
     public void run(){
           cerrojo.lock();
           try{
             synchronized(region_B){
                   System.out.println("hilo A");
                 }
           }finally {cerrojo.unlock();}
     }
   });

   Thread Hilo_B = new Thread(new Runnable(){
     public void run(){
           synchronized(region_B){
                 cerrojo.lock();
             try{
               System.out.println("hilo B");
                 }finally {cerrojo.unlock();}
           }
     }
   });

   Hilo_B.start();
   Hilo_A.start();
 }
}
```

a) **La aplicación puede imprimir los mensajes "hilo A" e "hilo B" o quedar bloqueada sin llegar a imprimir ningún mensaje.**
b) Primero se imprimirá el mensaje "hilo A" y a continuación la aplicación quedará bloqueada.
c) Primero se imprimirá el mensaje "hilo A" y a continuación el mensaje "hilo B".
d) La aplicación no compilará por no estar permitido anidar bloques de tipo synchronized y cerrojos de tipo ReentrantLock.

159 En una aplicación multihebrada escrita con C++ se desea controlar un recurso compartido entre hebras bajo control de exclusión mutua utilizando semáforos. En el momento de redactar la aplicación el estándar de C++ aún no incorpora esta técnica de control de la concurrencia. Para suplir la carencia, su equipo de programadores debe...

a) Esperar unos meses a que el nuevo estándar del lenguaje incorpore los semáforos.
b) **Escribir un monitor en C++ con una variable de condición y simular el semáforo con él.**
c) Escribir un monitor en C++ con dos variables de condición y simular el semáforo con él.
d) Escribir un monitor en C++ con una variable de condición y señalización SU, y simular el semáforo con él.

160 Considere el siguiente programa multihebrado escrito en Java. Indique el output que produce.

```
import java.util.concurrent.locks.*;

public class whatsUpDoc extends Thread{
    static whatsUpDoc [] h;
    ReentrantLock lock;
    Condition c;
    public whatsUpDoc() {lock=new ReentrantLock(); c=lock.newCondition();}

    public void run(){
        System.out.println("Hebra "+this.getName()+" saluda una...");
        lock.lock();
        try{c.await();
           System.out.println(this.getName()+" y otra vez...");
           h[9].notifyAll();
        } catch (InterruptedException e) {} finally {lock.unlock();}
    }

    public static void main(String[] args) throws InterruptedException{
        h = new whatsUpDoc[10];
        for(int i=0; i<10;i++){h[i]=new whatsUpDoc(); h[i].start();}
        for(int i=0; i<10;i++)h[i].join();
        System.out.print("Todos terminaron...");
    }
}
```

a) El programa se bloquea sin efectuar impresiones.
b) El programa imprime "Hebra Thread-i saluda una... y otra vez" con i variando entre 0 y 9 en cualquier orden, para terminar normalmente imprimiendo "Todos terminaron...".
c) El programa imprime "Hebra Thread-i saluda una... " con i variando entre 0 y 8 en cualquier orden,y después (y siempre) imprime "Hebra Thread-9 saluda una... " entrelazado con las impresiones ya citadas y queda bloqueado.
d) El programa imprime "Hebra Thread-i saluda una... " con i variando entre 0 y 9 en cualquier orden, y queda bloqueado.