Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final Teórico de la Asignatura
Junio de 2013

Apellidos: Nombre:

D.N.I.: Grupo (A ó B):

1. Notas

1. Escriba su nombre, apellidos, D.N.I. y grupo en el espacio habilitado para ello, y en todos los folios blancos que utilice. Firme el documento en la esquina superior derecha de la primera página.
2. Dispone de diez minutos para leer los enunciados y formular preguntas o aclaraciones sobre ellos. Transcurrido ese tiempo, no se contestarán preguntas. Dispone de 2:00 horas para completar el ejercicio.
3. No complete el documento a lápiz. Utilice bolígrafo o rotulador. Escriba con letra clara y legible. No podemos corregir lo que no se no puede leer.
4. Utilice los folios blancos que se le proporcionan para resolver los enunciados, pero traslade a este documento únicamente la solución final que obtenga, utilizando el espacio específicamente habilitado para ello, sin sobrepasarlo en ningún caso, y sin proporcionar información o respuestas no pedidas. Entregue tanto el enunciado como los folios blancos. Únicamente se corregirá este documento.

2. Criterios de Corrección

1. El examen se calificará de cero a diez puntos, y ponderará en la calificación final al 40 % bajo los supuestos recogidos en la ficha de la asignatura.
2. Cada enunciado incluye información de la puntuación que su resolución correcta representa, incluida entre corchetes.
3. Un enunciado (cuestión teórica o problema) se considera correcto si la solución dada es correcta completamente. En cualquier otro caso se considera incorrecto y no puntúa.

4. Un enunciado de múltiples apartados (cuestión teórica o problema) es correcto si y solo si todos los apartados que lo forman se contestan correctamente. En cualquier otro caso se considera incorrecto y no puntúa.

3. Cuestiones de Desarrollo Corto

Conteste a las preguntas que se le formulan en el espacio habilitado para ello. Deberá razonar o justificar su respuesta simpre que se le indique. La ausencia del razonamiento o de la justificación invalidará la respuesta al no ser esta completa.

1. [0.5 puntos] Como es sabido, un monitor garantiza por construcción la exclusión mutua de todos los procedimientos que define. Sin embargo, ¿deberían ser también reentrantes por construcción? Justifique su respuesta.

2. [0.5 puntos] Se desea pasar un semáforo como parámetro, y el lenguaje de soporte que se utiliza permite pasos por valor y referencia. ¿Cómo debería de pasarse? Justifique su respuesta.

3. [0.5 puntos] ¿Cuál es el principal inconveniente del uso de monitores en términos de rendimiento? Justifique su respuesta.

4. [1.0 puntos] Analice la corrección del algoritmo para control de la exclusión mutua expresado a continuación.

```
program em
var C1, C2: integer;
Process P1                    Process P2
begin                         begin
repeat                        repeat
 repeat C1:=1-C2               repeat C2:=1-C1
  until C2<>0;                  until C1<>0;
 Critica_1;                    Critica_2;
 C1:=1;                        C2:=1;
 Resto_1;                      Resto_2;
forever                       forever
end;                         end;

begin (*principal*)
 C1:=1; C2:=1;
 cobegin
  P1; P2;
 coend;
end.
```

Justifique su respuesta:

5. [1.0 puntos] Estudie el siguiente código en Java desde el punto de vista de la corrección parcial. Justique si es o no parcialmente correcto.

```
import java.util.concurrent.locks.*;
public class E24 extends Thread {
    static int x = 0;
    final static Lock cerrojo = new ReentrantLock();
    static Condition condicion = cerrojo.newCondition();
        public void run(){
        cerrojo.lock();
        try{
            while(x<5){ condicion.await(); }
            x++;
```

```
        }
        catch(Exception ex){} finally { cerrojo.unlock(); }
    }
    public static void main(String[] args) throws Exception{
        E24[] lista = new E24[5];
        for(int i=0;i<lista.length;i++){
            lista[i] = new E24();
            lista[i].start();}
        for(int i=0;i<lista.length;i++){
            lista[i].join();}
        System.out.print("Resultado: " + x);
  }
}
```

Justifique su respuesta:

6. [1.0 puntos] ¿Cual es la salida impresa del siguiente código Java?

```
import java.util.concurrent.CyclicBarrier;
import java.util.concurrent.atomic.AtomicInteger;

public class E23 extends Thread {
    static AtomicInteger x = new AtomicInteger(0);
    static CyclicBarrier barrera = new CyclicBarrier(5);
        public void run(){
        System.out.print(x.incrementAndGet());
        try{ barrera.await(); } catch(Exception ex){}
        System.out.print("a");
    }

    public static void main(String[] args) throws Exception{
        E23[] lista = new E23[5];
        for(int i=0;i<lista.length;i++){
            lista[i] = new E23();
            lista[i].start();}
        for(int i=0;i<lista.length;i++){
            lista[i].join();}
    }
}
```

Justifique su respuesta:

7. [1.0 puntos] Considere el siguiente código, que crea cuatro hilos, y observe el cuerpo del bucle de cada tipo de hilo, formado por una instrucción de asignación simple. Determine, de forma rigurosa y formal, qué tipos de hilos se pueden ejecutar concurrentemente y cuáles no.

```
public class ejecEntrelazada extends Thread{
 public static int a,b,c,w;
 public static int x, y, z;
 private int tipoHilo;
 public ejecEntrelazada(int tipoHilo){this.tipoHilo=tipoHilo;}
 public void run(){
   switch(tipoHilo){
   case 0:{for(int i=0; i<100000; i++){a=x+y;}; break;}
   case 1:{for(int j=0; j<100000; j++){b=z-1;};break;}
   case 2:{for(int k=0; k<100000; k++){c=a-b;};break;}
   case 3:{for(int h=0; h<100000; h++){w=c+1;};break;}
   }
 }

 public static void main(String[] args) throws Exception{
   a=0; b=0; c=0; w=0; x=1; y=1; z=1;
   ejecEntrelazada[] h = new ejecEntrelazada[4];
   for(int i=0; i<h.length;i++) h[i]=new ejecEntrelazada(i);
   for(int i=0; i<h.length;i++) h[i].start();
   for(int i=0; i<h.length;i++) h[i].join();
 }
}
```

Justifique su respuesta:

8. [1.0 puntos] Considere el siguiente código e indique cuál es la salida impresa que produce.

```
public class E30 extends Thread{
 private static Integer i = new Integer(1);
 private static int j=0;
 private E30 h;
 public E30(){j++;}
 public void run(){
   if(j<500)
     {h=new E30();
      synchronized(i){i++;}
      h.start();
      try{h.join();}catch (InterruptedException e){}
     }
 }

 public static void main(String[] args) throws Exception{
   E30 h = new E30();
   h.start();
   h.join();
   System.out.println(i.toString());
 }
}
```

Justifique su respuesta:

4. Problemas

1. [2 puntos] Un autómata celular unidimensional simplificado es una estructura $\langle A, \delta, n\rangle$ donde A es un array de componentes discretas llamadas células que toman valores un anillo abeliano finito (clase residual módulo n), Z_n; δ es una función de transición de la forma $x_i^{t+1} = \delta(x_{i-1}^t, x_i^t, x_{i+1}^t)$ que pemite recalcular el nuevo valor de una célula en tiempo $t+1$ en función de los valores de la propia célula y de sus vecinas en tiempo t; n define los estados posibles de cada celula en el espacio residual $Z_n = \{0, 1, 2, ..., n-1\}$. Considere un autómata concreto de 1000 células, con $n = 3$ y función de transición dada por:

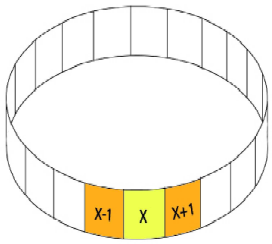

Figura 1: Autómata Celular con Condición de Frontera Cilíndrica

$$x_i^{t+1} = \delta(x_{i-1}^t, x_i^t, x_{i+1}^t) = (x_{i-1}^t + x_i^t + x_{i+1}^t) mod 3$$

El paso de un array al siguiente se conoce como el cálculo de una nueva generación de células, y el modelo puede ser utilizado en simulaciones discretas como contrapartida a modelos continuos basados en ecuaciones diferenciales ordinarias. El array tiene inicialmente a sus células en estados determinados de forma aleatoria, y las generaciones siguientes se calculan aplicando la función de transición célula por célula. Puesto que el modelo es finito, se aplica una condición de frontera que da a la estructura forma cilíndrica (ver la Figura 1).

Se desea escribir un programa multihebrado en java que permita calcular múltiples generaciones del autómata descrito por la función anterior. Para ello, debe determinar y explicar lo siguiente:

- ¿Qué coeficiente de bloqueo C_b presentará una solución multihebrada?¿Por qué?

- A partir del valor de C_b anterior, y supuesto que $N_{nd} = 4$ ¿Cuánto vale N_t?

- ¿Cómo particionará el problema para lograr paralelismo *multicore*?

- Escriba una clase `aCelularParalelo` que herede de `Thread` y que dé soporte a ese paralelismo.

- Escriba un programa principal que lea el número de generaciones y realize la computación del autómata celular.

2. [1.5 puntos] Considere una fragata con un sistema de lanzamiento de misiles con dos plataformas A y B a proa y popa. Los artilleros del buque son de tercera, segunda y primera clase. Los artilleros de tercera clase usan la plataforma A, los de segunda la B, y los de primera -auténticos especialistas- ambas de forma simultánea. Desarrolle un código que modele el proceso de cada clase de artillero para tomar y liberar las plataformas, utilizando semáforos teóricos *Dijsktra*. Se supone que los artilleros intentan acceder simultáneamente a las plataforma, puesto que utilizan sistemas concurrentes de adquisión de blancos.-

(continúe escribiendo aquí su solución al problema número 2)