Apellidos: **Nombre:** **Matrícula:**

Prueba objetiva 2 - Clave a

Concurrencia

2011-2012 - Primer semestre

Lenguajes, Sistemas Informáticos e Ingeniería de Software

Normas

Este es un cuestionario que consta de **4 preguntas** en **3 páginas**. Todas las preguntas son **preguntas de respuesta simple** excepto la pregunta 4 que es una **pregunta de desarrollo**. La puntuación total del examen es de **8 puntos**. La duración total es de **una hora**. El examen debe contestarse en las **hojas de respuestas**. No olvidéis rellenar **apellidos, nombre y DNI** en cada hoja de respuesta.

Sólo hay una respuesta válida a cada pregunta de respuesta simple. Toda pregunta en que se marque más de una respuesta se considerará incorrectamente contestada. Toda pregunta incorrectamente contestada restará del examen una cantidad de puntos igual a la puntuación de la pregunta dividido por el número de alternativas ofrecidas en la misma.

Cuestionario

(2 puntos) 1. Un recurso que encapsula una cuenta bancaria compartida va a ser implementado con métodos *synchronized*. La operación *Reintegro* se especifica de la manera siguiente:

CPRE: $self.Saldo >= c$
Reintegro(c)
POST: $self.Saldo = self^{pre}.Saldo - c$

La operación *Reintegro* se ha implementado mediante el código siguiente:

```
public synchronized void reintegro(int c) {
   // traduccion CPRE
   if (c > this.saldo) {
     try {
       wait();
     } catch (Exception e) {}
   }
   // traduccion POST
   this.saldo = this.saldo - c;
   // codigo desbloqueos
   notifyAll();
}
```

Se pide señalar la respuesta correcta:

(a) Es una implementación correcta de la operación *Reintegro* usando métodos *synchronized*, `wait()` y `notifyAll()`.

(b) Ese código puede provocar que se ejecute una operación cuya CPRE no se cumple.

(2 puntos) 2. Supóngase que una condición de sincronización (*CPRE*) de una operación *Op* de un recurso compartido depende del estado del recurso y de un parámetro de entrada (x). Supóngase que dicho recurso va a ser implementado con monitores y que la operación va a ser llamada a lo sumo por un único proceso.

Se pide señalar la respuesta correcta:

(a) Para implementar la sincronización condicional de *Op* es necesario crear una variable `Cond` por cada posible valor de x.

(b) Es posible implementar la sincronización condicional de *Op* con una única variable `Cond`.

(2 puntos) 3. Dado el siguiente programa implementado con la librería JCSP:

```
static private Any2OneChannel c1 = Channel.any2one();
static private Any2OneChannel c2 = Channel.any2one();
```

```
static class A implements CSProcess {
    public void run() {
        c1.out().write("A");
        String s =
            (String) c2.in().read();
        System.out.print(s);
    }
}
```

```
static class B implements CSProcess {
    public void run() {
        String s =
            (String) c1.in().read();
        c2.out().write("B");
        System.out.print(s);
    }
}
```

```
    // Programa principal
    CSProcess sistema = new Parallel(new CSProcess[] {new A(), new B()});
    sistema.run();
```

Se pide: señalar la respuesta correcta.

(a) La salida del programa es `AB` o `BA`.

(b) La salida del programa es siempre `AB`.

(c) La salida del programa es siempre `BA`.

(d) Se produce un interbloqueo y el programa no produce salida alguna.

Apellidos: **Nombre:** **Matrícula:**

(4 puntos) 4. Supongamos un programa concurrente con procesos (al menos uno) que ejecutan repetidamente operaciones $r.inc(x)$ con x positivo y $x < N/2$ y procesos (al menos uno) que ejecutan repetidamente operaciones $r.dec(y)$ con y positivo e $y < N/2$, siendo r un recurso compartido del tipo especificado a continuación:

C-TAD MultiCont
OPERACIONES
ACCIÓN inc: $\mathbb{N}$*[e]*
ACCIÓN dec: $\mathbb{N}$*[e]*
SEMÁNTICA
DOMINIO:
TIPO: *MultiCont* = $\mathbb{N}$
INVARIANTE: $0 \leq \text{self} \wedge \text{self} \leq N$

INICIAL: $\text{self} = 0$

PRE: $n > 0 \wedge n < N/2$
CPRE: $\text{self} + n \leq N$
inc(n)
POST: $\text{self} = \text{self}^{pre} + n$

PRE: $n > 0 \wedge n < N/2$
CPRE: $n \leq \text{self}$
dec(n)
POST: $\text{self} + n = \text{self}^{pre}$

Se pide implementar el recurso compartido en Java utilizando como mecanismo de concurrencia las clases `Monitor` y `Monitor.Cond` de la librería `es.upm.babel.cclib`.

Nota: se pueden ignorar los problemas de inanición a la hora de implementar, es decir, la implementación deberá bloquear a cualquier proceso que invoque una operación cuya CPRE no se cumple y al terminar de ejecutar una operación deberá desbloquearse un proceso (no importa cuál) siempre y cuando algún proceso pueda ser desbloqueado.

(Responder en esta y en la siguiente página).