**Apellidos:** **Nombre:** **Matrícula:**

# Prueba final, parte 1 - Clave a
## Concurrencia
**2011-2012 - Primer semestre**
**Dpto. de Lenguajes, Sistemas Informáticos e Ingeniería de Software**

## Normas

Este es un cuestionario que consta de **4 preguntas** en **4 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 **10 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. Dado el siguiente **CTAD** (sólo se muestran las partes necesarias):

**TIPO:** T_Cuatro = $(a:\mathbb{B} \times b:\mathbb{B})$
**INICIAL:** $self = (falso, falso)$
**CPRE:** *cierto*
**S()**
**POST:** $self^{pre} = (pe, se) \wedge self = (ps, \neg se) \wedge ps = (\neg pe \wedge se) \vee (pe \wedge \neg se)$
**CPRE:** $self \neq (cierto, falso)$
**M()**
**POST:** $self = (\neg self^{pre}.a, self^{pre}.b)$

Se pide marcar la afirmación correcta:

(a) El recurso puede pasar, a lo sumo, por 3 estados diferentes.

(b) El recurso podría estar en un estado dado y, tras ejecutarse una sola de sus acciones, continuar en el mismo estado.

(c) Si el sistema solo contiene (además de un recurso de este tipo) un único proceso que intenta ejecutar $S$ indefinidamente, el sistema podría acabar en interbloqueo.

(d) Si el sistema consta de un recurso de este tipo, un proceso que invoca a $S$ una sola vez y otro que intenta ejecutar $M$ indefinidamente, el sistema no puede terminar en interbloqueo.

(Sugerencia: Dibuja aquí el grafo de los estados por los que puede pasar el recurso.)

(2 puntos) 2. Dado un programa concurrente en la que tres *threads* instancias de las clases `C`, `D` y `E` comparten una variable `n`:

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

static class C extends Thread {
    public void run() {
        s2.acquire();
        s1.acquire();
        n = 2 * n;
        s1.release();
    }
}
```

```
static class D extends Thread {
    public void run() {
        s1.acquire();
        n = n * n;
        s1.release();
    }
}

static class E extends Thread {
    public void run() {
        s1.acquire();
        n = n + 3;
        s2.release();
        s1.release();
    }
}
```

**Se pide** marcar el conjunto que contiene únicamente los posibles valores de la variable `n` tras la terminación de los tres threads.

(a) $\{4, 16, 32, 64\}$

(b) $\{4, 8, 32, 64\}$

(c) $\{8, 32, 64\}$

(d) $\{5, 8, 32\}$

(2 puntos) 3. Dado el siguiente programa concurrente

```
static int x = 0;
```

```
static class T extends Thread {
    private int y;
    public T (int y) {
        this.y = y;
    }
```

```
    public void run() {
        int z = y;
        z = z + y; x = x + z;
    }
}
```

```
    // Programa principal
    Thread[] t = new Thread[] {new T(1), new T(2)};
    t[0].start(); t[1].start(); t[0].join(); t[1].join();
```

**Se pide** marcar la afirmación correcta.

(a) Es necesario asegurar exclusión mutua en el acceso a la variable `x`.

(b) Es necesario asegurar exclusión mutua en el acceso al atributo `y`.

(c) Es necesario asegurar exclusión mutua en el acceso a la variable `z`.

(d) No es necesario asegurar exclusión mutua en el acceso a ninguna de esas tres variables.

(4 puntos) 4. Está a punto de inaugurarse el primer centro comercial en el que la compra la realizan robots: *e*-QEA. Los compradores envían sus listas de la compra y sus robots inician el recorrido. *e*-QEA lo componen un número fijo ($N$) de secciones adyacentes. Los robots inician la compra entrando en la sección 0, avanzan a la sección adyacente (0, 1, ...) cuando han acumulado los productos requeridos de esa sección y terminan su compra saliendo de la sección $N-1$.

El centro *e*-QEA tiene un problema estructural (literalmente): cada sección soporta un peso máximo ($P$). Esto significa que cuando un robot con un determinado peso ($p$) quiere avanzar, debe esperar hasta que el incremento de peso que provoca no ponga en peligro la estructura (si el peso actual de la sección adyacente es *peso* entonces un robot con peso $p$ no puede avanzar si $peso + p > P$).

Se desea crear un recurso compartido para gestionar el movimiento de robots por *e*-QEA. Los *threads* que controlan los robots ejecutan la operación $avanzar(s, p)$ cuando el robot quiere avanzar de la sección $s-1$ a la sección $s$ portando un peso $p$ (con $s = 0$ se indica la entrada al centro y con $s = N$ la salida del mismo).

Se pide completar la especificación de un recurso compartido *Controle*-QEA que controle el avance de los robots por el centro *e*-QEA:

**C-TAD** Control*e*-QEA

**OPERACIONES**

**ACCIÓN** avanzar: *TipoSección[e]* × ℕ*[e]*

**SEMÁNTICA**

**DOMINIO:**

**TIPO:** *Controle*-QEA =

**TIPO:** *TipoSección = {0, 1, ..., N}*

**INVARIANTE:**

**INICIAL:**

**PRE:**

**CPRE:**

**avanzar(s,p)**

**POST:**

(Página intencionadamente en blanco, puede usarse como hoja en sucio).