Apellidos: **Nombre:** **N. Matrícula:**

Concurrencia (parte 2)/clave: b

Curso 2013-2014 - 1er. semestre (enero 2014)
Grado en Ingeniería Informática/Grado en Matemáticas e Informática
UNIVERSIDAD POLITÉCNICA DE MADRID

NORMAS: Este es un cuestionario que consta de 5 preguntas. Todas las preguntas son de respuesta simple excepto la pregunta 1 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 número de matrícula 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.

(5 puntos) 1. En un programa concurrente hay dos tipos de procesos que acceden a una base de datos: los procesos lectores y los procesos escritores. Muchos procesos pueden leer de la base de datos simultáneamente pero cuando un proceso quiere escribir sobre la base de datos ningún otro proceso puede realizar ninguna operación sobre la misma.

Para asegurar que el programa cumple esta propiedad, se ha diseñado un recurso compartido que controla el acceso de los procesos a la base de datos. Los procesos lectores ejecutan la operación *IL* antes de leer y la operación *FL* después de hacerlo mientras que los procesos escritores ejecutan la operación *IE* antes de escribir y la operación *FE* después de hacerlo.

La especificación formal de dicho recurso compartido es la siguiente:

TIPO: $LE = (l : \mathbb{Z} \times e : \mathbb{Z})$
INVARIANTE: $\forall\, r \in LE \bullet\ r.e \geq 0 \wedge r.l \geq 0 \wedge r.e \leq 1 \wedge ((r.e > 0 \Rightarrow r.l = 0) \wedge (r.l > 0 \Rightarrow r.e = 0))$
INICIAL: self $= (0,0)$
CPRE: self.$e = 0$
IL()
POST: self $= (\text{self}^{pre}.l + 1, \text{self}^{pre}.e)$
CPRE: self.$e = 0 \wedge$ self.$l = 0$
IE()
POST: self $= (\text{self}^{pre}.l, \text{self}^{pre}.e + 1)$
CPRE: *cierto*
FL()
POST: self $= (\text{self}^{pre}.l - 1, \text{self}^{pre}.e)$
CPRE: *cierto*
FE()
POST: self $= (\text{self}^{pre}.l, \text{self}^{pre}.e - 1)$

Se pide implementar dicho recurso en Java usando la librería `Monitor`.

(1½ puntos) 2. Supóngase un objeto `c` de una clase `C` en la que se han definido los métodos `o` y `p`, declarados como **`synchronized`**. Supóngase también que un thread `t` entra en un bucle infinito *mientras* ejecuta el código de `c.o()`.

Se pide señalar la respuesta correcta.

(a) Cualquier otro thread distinto de `t` que intente ejecutar `c.p()` no quedará bloqueado porque la exclusión mutua no se tiene en cuenta en el caso de un bucle infinito.

(b) Cualquier otro thread distinto de `t` que intente ejecutar `c.p()` no quedará bloqueado porque la exclusión mutua afecta sólo a las llamadas a `c.o()`.

(c) Cualquier otro thread distinto de `t` que intente ejecutar `c.p()` quedará bloqueado para asegurar la exclusión mutua.

(1½ puntos) 3. Supóngase que dos threads ejecutan los siguientes bucles:

```
while (true) {
   S11;
   ch1.in.read();
   S12;
   ch2.out.write(null);
   S13;
}
```

```
while (true) {
  ch1.out.write(null);
  ch2.in.read();
  S2;
}
```

Suponemos que no hay otros threads que afecten al problema aparte de los mostrados. **Se pide** señalar la respuesta correcta.

(a) `S13` y `S2` no pueden ejecutar simultáneamente.

(b) `S12` y `S2` no pueden ejecutar simultáneamente.

(c) `S11` y `S2` no pueden ejecutar simultáneamente.

(1 punto) 4. Dado el siguiente código:

```
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() {
        c2.out().write("B");
        String s =
            (String) c1.in().read();
        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 siempre `BA`.

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

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

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

(1 punto) 5. 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) Es posible implementar la sincronización condicional de *Op* con una única variable `Cond`.

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

[Escribe aquí la solución a la pregunta 1.]