Apellidos: **Nombre:** **Matrícula:**

Concurrencia (parte 1)/clave: a

Curso 2018–2019 - 2º semestre (julio 2019)
Grado en Ingeniería Informática / Grado en Matemáticas e Informática /
Doble Grado en Ing. Informática y ADE
UNIVERSIDAD POLITÉCNICA DE MADRID

NORMAS: Este es un cuestionario que consta de 7 preguntas. Todas las preguntas son de respuesta simple excepto la pregunta 7 que es una pregunta de desarrollo. La puntuación total del examen es de 10 puntos. La duración total es de una hora. No olvidéis rellenar vuestros datos.

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

($1\frac{1}{2}$ puntos) 1. Dado el siguiente programa:

```
class Carreras {
  private static volatile int a = 0;

  static class Hilo extends Thread {

    private static volatile int b = 0;

    private volatile int c = 0;

    public void run () {
      a++;
      b++;
      c++;
      System.out.print(c);
    }
  }
```

```
  public static final void
    main(final String[] args)
    throws Exception {

    Thread t1 = new Hilo();
    Thread t2 = new Hilo();

    t1.start();
    t2.start();

    System.out.print("ejecutando");

    t1.join();
    t2.join();
  }
}
```

Se pide marcar la afirmación correcta.

(a) La sentencia `a++;` no es una condición de carrera.

(b) La sentencia `b++;` no es una condición de carrera.

(c) La sentencia `c++;` no es una condición de carrera.

(1 punto) 2. Dado el programa de la pregunta anterior. **Se pide** marcar la afirmación correcta.

(a) `"ejecutando11"` es una salida posible del programa.

(b) `"ejecutando11"` no es una salida posible del programa.

(1 punto) 3. Dado siguiente **CTAD** (el parámetro de la operación *peligro* es un dato de entrada de tipo $\mathbb{B}$).

C-TAD *Peligro*
TIPO: *Peligro* $= p : \mathbb{B} \times o : \mathbb{N}$
INICIAL: self $= (\mathit{false}, 0)$

INVARIANTE: self.$p \vee$ self.$o \leq 5$

CPRE: Cierto
***peligro*(x)**
POST: self.$p = x \wedge$ self.$o =$ self.o

CPRE: $\neg$self.$p \wedge$ self.$o < 5$
***entrar*()**
POST: $\neg$self.$p \wedge$ self.$o =$ self.$o + 1$

CPRE: self.$o > 0$
***salir*()**
POST: self.$p =$ self.$p \wedge$ self.$o =$ self.$o - 1$

Se pide marcar la afirmación correcta.

(a) Siempre se cumplirá la invariante.

(b) Podría llegar a violarse la invariante.

(1½ puntos) 4. Dada la siguiente implementación del **CTAD** de la pregunta anterior:

```
class Peligro {
  private Semaphore p =
    new Semaphore(1);
  private Semaphore o =
    new Semaphore(5);

  public void peligro(boolean x) {
    if (x)
      o.signal();
    else
      o.await();
  }
  public void entrar() {
    o.await();
    p.await();
    p.signal();
  }

  public void salir() {
    o.signal();
  }
}
```

Supóngase se realizan simultáneamente las siguientes siete llamadas por sendos procesos: `peligro(true)`, `entrar()`, `entrar()`, `entrar()`, `entrar()`, `entrar()` y `entrar()` y supóngase que no se realiza ninguna otra llamada al recurso compartido.

Se pide marcar la afirmación correcta.

(a) Ningún proceso estará bloqueado.

(b) Un proceso estará bloqueado.

(c) Más de un proceso estará bloqueado.

(1 punto) 5. Dado un programa concurrente en la que tres *threads* instancias de las clases `A`, `B` y `C` comparten las variables `x`, `y`, `sx` y `sy`.

```
static int x = 1;
static int y = 2;
static Semaphore sx = new Semaphore(0);
static Semaphore sy = new Semaphore(0);
```

```
class A extends Thread {
  public void run() {
    x = x + 1;
    sx.signal();
  }
}
```

```
class B extends Thread {
  public void run() {
    y = y + 1;
    sy.signal();
  }
}
```

```
class C extends Thread {
  public void run() {
    sx.await();
    sy.await();
    System.out.print(x+y);
  }
}
```

Se pide marcar la afirmación correcta.

(a) `"5"` es una salida posible del programa.

(b) `"5"` no es una salida posible del programa.

(1 punto) 6. Si en el código anterior el semáforo `sx` se inicializa a 1. **Se pide** marcar la afirmación correcta.

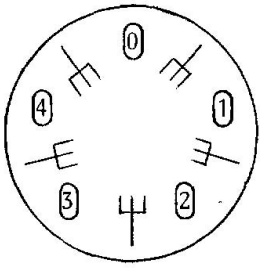
(a) `"5"` es una salida posible del programa.

(b) `"5"` no es una salida posible del programa.

Apellidos: **Nombre:** **Matrícula:**

(3 puntos) 7. Lo que veis a continuación es la descripción original de Edsger W. Dijkstra del problema de los *Dining Philosophers*:

> *La vida de un filósofo consiste en alternar pensar y comer en una especie de bucle infinito. Cinco filósofos, numerados del 0 al 4, viven en una casa con una mesa en la que cada filósofo tiene su sitio asignado:*



> *Su único problema, al margen de los filosóficos, es que su menú consiste en un tipo complicado de espagueti que es necesario comer con dos tenedores. Hay un único tenedor a cada lado de cada plato. Eso no es un problema. Sin embargo, como consecuencia, no puede haber dos filósofos comiendo a la vez uno al lado del otro.*

Nosotros hemos decidido representar el problema en forma de un recurso compartido *m* (de mesa) con dos operaciones: *m.cogerTenedores*(i) y *m.soltarTenedores*(i). Esas serán las operaciones que el filósofo *i* ejecutará antes y después de comer, de forma que tendrá que bloquear en *m.cogerTenedores*(i) cuando alguno de los tenedores a los lados de su plato estén siendo usados (por el filósofo $(i+1) \bmod 5$ o por el filósofo $(i-1) \bmod 5$).

Se pide completar el recurso compartido sobre el esqueleto en la siguiente página.

Nota: se sugiere que el dominio del recurso sea una función parcial que hable de los tenedores libres. Las funciones parciales formalizan las tablas. La declaración del tipo de una función parcial de *A* en *B* es $A \nrightarrow B$. Si self es una función parcial del tipo $A \nrightarrow B$, la expresión self(a) representa el valor de para la *clave a*, la expresión self $\oplus \{a \mapsto b\}$ representa la *tabla* self cambiando la entrada de la *clave a* por el valor *b* y la expresión dom self representa el conjunto de claves de self.

C-TAD Mesa

OPERACIONES

ACCIÓN cogerTenedores: $\mathbb{N}$*[e]*
ACCIÓN soltarTenedores: $\mathbb{N}$*[e]*

SEMÁNTICA
DOMINIO:

TIPO: *Mesa* =

INVARIANTE:

INICIAL:

PRE:

CPRE:

cogerTenedores(i)

POST:

PRE:

CPRE:

soltarTenedores(i)

POST: