Apellidos: **Nombre:** **Matrícula:**

Concurrencia (parte 2)/clave: a

Curso 2018–2019 - convocatoria extraordinaria (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 5 preguntas. Todas las preguntas son de respuesta simple excepto la pregunta 5 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

(1½ puntos) 1. 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*) que puede tomar dos valores. 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) No es posible implementar la sincronización condicional de *Op* con una única variable `Cond`.

(b) Es posible implementar la sincronización condicional de *Op* con dos variables `Cond`.

(1½ puntos) 2. Supóngase el siguiente código:

```
Monitor m = new Monitor();
Cond c = m.newCond();
```

Se pide marcar la afirmación correcta.

(a) Al ejecutar `c.await()` se libera el monitor `m`.

(b) Es necesario ejecutar `m.leave()` inmediatamente después de ejecutar `c.await()` para liberar el monitor `m`.

(1½ puntos) 3. Dado el siguiente programa concurrente que ejecuta dos procesos JCSP con paso de mensajes síncrono:

```
static One2OneChannel c = Channel.one2one();
```

```
class A implements CSProcess {
  public void run() {
    System.out.print("1");
    c.out().write(null);
    System.out.print("2");
  }
}
```

```
class B extends Thread {
  public void run() {
    System.out.print("3");
    c.in().read();
    System.out.print("4");
  }
}
```

Se pide marcar la afirmación correcta.

(a) `"1234"` es una salida posible del programa.

(b) `"1234"` no es una salida posible del programa.

(1½ puntos) 4. A continuación mostramos la especificación formal del recurso compartido *Peligro*. Inmediatamente después mostramos una implementación del mismo utilizando JCSP:

C-TAD *Peligro*
OPERACIONES
ACCIÓN *avisarPeligro*: $\mathbb{B}[e]$
ACCIÓN *entrar*:
ACCIÓN *salir*:

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

INVARIANTE: self.$o \leq 5$

CPRE: Cierto
***avisarPeligro*(x)**
POST: self.$p = x \wedge$ self.$o =$ self^{pre}.o

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

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

```
class PeligroCSP
  implements CSProcess {
  private Any2OneChannel cp =
    Channel.any2one();
  private Any2OneChannel ce =
    Channel.any2one();
  private Any2OneChannel cs =
    Channel.any2one();

  public PeligroCSP() {
    new ProcessManager(this).start();
  }

  public void avisarPeligro(boolean x)
  {
    cp.out().write(x);
  }

  public void entrar() {
    ce.out().write(null);
  }

  public void salir() {
    cs.out().write(null);
  }
```

```
  public void run() {
    Boolean p = false;
    Integer o = 0;
    Guard[] inputs =
      {cp.in(), ce.in(), cs.in()};
    Alternative services =
      new Alternative(inputs);
    while(true) {
      switch(services.fairSelect()) {
      case 0:
        p = (Boolean)cp.in().read();
        break;
      case 1:
        ce.in().read();
        if (!p && o < 5)
          o++;
        break;
      case 2:
        cs.in().read();
        if (o > 0)
          o--;
        break;
      }
    }
  }
}
```

Se pide marcar la afirmación correcta.

(a) Es una implementación correcta del recurso compartido.

(b) No es una implementación correcta del recurso compartido.

Apellidos: **Nombre:** **Matrícula:**

(4 puntos) 5. Dada la especificación formal de un recurso *Peligro* del ejercicio 4. **Se pide:** Completar la implementación de este recurso mediante monitores.[1]

```
class Peligro {
  // Estado del recurso (inicializacion incluida)
```

```
boolean p;
int o=0;
```

```
  // Monitores y conditions (inicializacion incluida)
```

```
Monitor mutex = new Monitor();
Monitor.Cond c1 = mutex.newCond();
"       "    c2 = "       "    c1
```

```
  public Peligro() {
  }
  public void avisarPeligro(boolean x) {
```

```
mutex.enter();

this.p = x;
desbloquear();
mutex.leave();
```

```
  }
  public void entrar() {
```

```
mutex.enter();
if( p && o>=5) c1.await;

  o++;
 p = !p;

desbloquear

mutex.leave();
```

```
  }
  public void salir() {
```

```
mutex.enter();
if(o<=0) c2.await();

  o--;

  desbloquear();

mutex.leave();
```

```
  }
```

[1] En la siguiente página puedes implementar un método privado de desbloqueo si no te atreves a optimizar el desbloqueo.

```
    private void desbloquear() {
```

```
        if(!p && o<S) c1.signal();
        else if (o>0) c2.signal();
```

```
    }
}
```