Apellidos: **Nombre:** **Matrícula:**

Concurrencia (parte 2)/clave: b

Curso 2018–2019 - 2º Semestre (mayo 2019)

Respuestas

1	2	3	4	5	6

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. El test debe contestarse en la **caja de respuestas**. No olvidéis rellenar **apellidos, nombre y número de matrícula** en cada hoja.

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. Considere el siguiente código:

```
class P1
  implements CSProcess{
  public void run(){
    ch1.out().write(null);
    System.out.print("A");
    ch3.in().read();
  }
}
```

```
class P2
  implements CSProcess{
  public void run(){
    ch1.in().read();
    System.out.print("C");
    ch2.out().write();
  }
}
```

```
class P3
  implements CSProcess{
  public void run(){
    System.out.print("B");
    ch2.in().read();
    ch3.out().write(null);
  }
}
```

```
    Any2OneChannel ch1 = Channel.any2one();
    Any2OneChannel ch2 = Channel.any2one();
    Any2OneChannel ch3 = Channel.any2one();
    public static final void main(final String[] args){
        new Parallel (new CSProcess[] {new P1(), new P2(), new P3()}).run();
    }
```

Se pide señalar la respuesta correcta.

(a) La salida del programa sólo podrá ser `ACB` o `CAB`.

(b) Se imprimirá `A` tras lo cual el programa quedará bloqueado.

(c) La salida del programa siempre será `CAB`.

(d) El programa siempre acabará, pero no sabemos con qué salida en concreto.

(1 punto) 2. Dado el siguiente **CTAD**:

ACCIÓN op1:
ACCIÓN op2:

TIPO: MiCTAD $= \mathbb{Z}$
INICIAL: *self* $= 0$

CPRE: *self* $<= 0$
op1()
POST: self $=$ self$^{pre} + 1$
CPRE: self $>= 0$
op2()
POST: self $= -1$

Se ha decidido implementarlo con monitores mediante el siguiente código:

```
public class MiCTAD{
  private Monitor mutex =
    new Monitor();
  private Monitor.Cond c1 =
    mutex.newCond();
  private Monitor.Cond c2 =
    mutex.newCond();
  private int self = 0;

  public void op1(){
    mutex.enter();
    if (self > 0) {c1.await();}
    self ++;
    c2.signal();
    mutex.leave();
}
```

```
  public void op2(){
    mutex.enter();
    if (self < 0) {c2.await();}
    self = -1;
    c1.signal();
    mutex.leave();
}
```

Se pide marcar la afirmación correcta:

(a) Podría darse el caso de que hubiese hilos esperando en `c1` o en `c2` que, pudiendo ejecutarse, no se desbloqueen.

(b) Se trata de una implementación correcta del recurso compartido

(c) Podrían ejecutarse operaciones cuya CPRE no se cumple.

(1 punto) 3. Dada esta otra implementación del CTAD del ejercicio anterior.

```
public class MiCTAD{
  private Monitor mutex =
    new Monitor();
  private Monitor.Cond c1 =
    mutex.newCond();
  private Monitor.Cond c2 =
    mutex.newCond();
  private int self = 0;

  public void op1(){
    mutex.enter();
    if (self > 0) {c1.await();}
    self ++;
    desbloquear();
    mutex.leave();
}
```

```
  public void op2(){
    mutex.enter();
    if (self < 0) {c2.await();}
    self = -1;
    desbloquear();
    mutex.leave();
}


private void desbloquear () {
  if (self <= 0) {
    c1.signal();
  }
  else if (self >= 0) {
    c2.signal();
  }
}
```

Se pide señalar la respuesta correcta.

(a) Podría darse el caso de que hubiese hilos esperando en `c1` o en `c2` que, pudiendo ejecutarse, no se desbloqueen.

(b) Se trata de una implementación correcta del recurso compartido

(c) Podría ejecutarse procesos cuya CPRE no se cumple

(1 punto) 4. Supóngase que una condición de sincronización (*CPRE*) de una operación *Op* de un recurso compartido depende del estado del recurso y de dos parámetros de entrada: x, que puede tomar 3 valores e y, que puede tomar 7 valores. Supóngase que dicho recurso va a ser implementado con monitores, siguiendo la metodología vista en clase, y que la operación va a ser invocada por un número indefinido (> 100) de procesos. **Se pide** señalar la respuesta correcta:

(a) Es posible implementar la sincronización condicional de *Op* con 10 objetos `Cond`.

(b) Es posible implementar la sincronización condicional de *Op* con 21 objetos `Cond`.

(c) Es posible implementar la sincronización condicional de *Op* con un solo objeto `Cond`.

Apellidos: **Nombre:** **Matrícula:**

(1½ puntos) 5. Se define una clase *servidor* `P` y tres clases *cliente* `P1`, `P2` y `P3` que se comunican a través de los canales de comunicación `petA`, `petB` y `petC` (se muestran las partes relevantes del código para este problema).

```
class P implements CSProcess {
  public void run() {
    boolean q = true;
    boolean r = true;
    boolean s = false;
    final int A = 0;
    final int B = 1;
    final int C = 2;
    final Guard[] entradas =
        {petA.in(), petB.in(), petC.in()};
    final Alternative servicios =
        new Alternative (entradas);
    final boolean[] sincCond =
        new boolean[3];

    while (true) {
      sincCond[A] = (q && r);
      sincCond[B] = (r && s);
      sincCond[C] = (s && q);
      int sel = servicios.fairSelect(sincCond);
      switch (sel) {
      case A:
        petA.in().read();
        q = !q;
        s = !s;
        System.out.println("A");
        break;
      case B:
        petB.in().read();
        r = !r;
        q = !q;
        System.out.println("B");
        break;
      case C:
        petC.in().read();
        s = !s;
        r = !r;
        System.out.println("C");
        break;
      }
    }
  }
}
```

```
class P1 implements CSProcess {
  public void run() {
    while (true) {
      petA.out().write(null);
    }
  }
}

class P2 implements CSProcess {
  public void run() {
    while (true) {
      petB.out().write(null);
    }
  }
}

class P3 implements CSProcess {
  public void run() {
    while (true) {
      petC.out().write(null);
    }
  }
}
```

Dado un programa concurrente con tres procesos `p`, `p1`, `p2` y `p3` de las clases `P`, `P1`, `P2` y `P3`.

Se pide marcar cuál de las siguientes afirmaciones es la correcta:

(a) El programa ejecutará indefinidamente y la salida por consola será `ABCABCABC...` indefinidamente.

(b) El programa imprimirá `AB` y los 4 procesos acabarán bloqueados.

(c) Es seguro que los cuatro procesos van a ejecutar indefinidamente, pero no se puede saber con qué salida concreta.

(d) Ninguna de las otras respuestas.

(1 punto) 6. Dado un recurso compartido implementado con JCSP.

Se pide decir cuál de las siguientes afirmaciones es la correcta:

(a) Cuando `fairSelect(sincCond)` devuelve un valor `sel` sabemos que hay una petición en el canal que ocupa la posición `sel` del array de alternativas y que `sincCond[sel] == true`.

(b) Cuando `fairSelect(sincCond)` devuelve un valor `sel` no sabemos si habrá alguna petición en el canal que ocupa la posicion `sel` y la lectura sobre dicho canal se podrá quedar bloqueada.

(3 puntos) 7. A continuación mostramos la especificación formal de un recurso gestor *Misil*.

C-TAD Misil
OPERACIONES
ACCIÓN notificar: $\mathbb{Z}[e]$
ACCIÓN detectarDesviacion: $TUmbral[e] \times \mathbb{Z}[s]$

SEMÁNTICA
DOMINIO:
TIPO: $TUmbral = [0..100]$
TIPO: $Misil = \mathbb{Z}$

INICIAL: self $= 0$

CPRE: *Cierto*
notificar(desv)
POST: self $=$ *desv*

CPRE: $\mid$ self $\mid >$ *umbral*
detectarDesviacion(umbral,d)
POST: *self* $=$ self$^{pre} \wedge d =$ self^{pre}

Se pide: Completar la implementación de este recurso mediante monitores. En cuanto al código de desbloqueos podéis optar tanto por un método de desbloqueo genérico como por tener código de desbloqueo especializado en los distintos métodos. Si optáis por la segunda posibilidad dejad en blanco el cuerpo del método `desbloqueo`. NOTA: Podéis usar el método `Math.abs(x)` para calcular el valor absoluto de un número, $\mid x \mid$.

Apellidos: **Nombre:** **Matrícula:**

```
class Misil  {
  // estado del recurso
```

```
  // Declaramos monitores y colas condition
```

```
  public Misil() {
```

```
  }

  public void notificar(int desv) {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

```
    // codigo de desbloqueo y salida de la seccion critica
```

```
  }
  public int detectarDesviacion(int umbral) {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

```
    // codigo de desbloqueo, salida de la seccion critica y return
```

```
  }

  private void desbloqueo() {
```

```
  }
}
```