Apellidos: **Nombre:** **Matrícula:**

Concurrencia (parte 2)/clave: a

Curso 2015–2016 - 2º Semestre (Mayo 2016)
Grado en Ingeniería Informática / Grado en Matemáticas e Informática
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 examen debe contestarse en las **mismas hojas**. No olvidéis rellenar **apellidos, nombre y número de matricula** 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 punto) 1. Considere el siguiente código:

```
class P1
  implements CSProcess{
  public void run(){
    One2OneChannel chResp =
      Channel.one2one();
    ch1.out().write(
      chResp.out());
    chResp.in().read();
    System.out.print("A");
  }
}
```

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

```
class P3
  implements CSProcess{
  public void run(){
    ChannelOutput resp =
      (ChannelOutput)
        ch2.in().read();
    resp.write(null);
    System.out.print("C");
  }
}
```

```
    Any2OneChannel ch1 = Channel.any2one();
    Any2OneChannel ch2 = 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 siempre será `ABC`.

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

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

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

(1 punto) 2. Suponed que en el código anterior cambiamos los canales **ch1 y ch2** por canales del tipo **One2OneChannel** haciendo todos los cambios que sean necesarios para ello.

Se pide señalar la respuesta correcta.

(a) Nos dará un error cuando lo compilemos.

(b) Seguirá funcionando igual que antes.

(c) Nos dará un error en tiempo de ejecución.

(1 punto) 3. ¿Debería permitirse a un *thread* invocar una operación de `await` sobre un objeto de clase `Monitor.Cond` generado a partir de un objeto de la clase `Monitor` **sin previamente haber invocado** el método `enter`?

(a) Sí.

(b) No.

(1½ puntos) 4. Dado el siguiente **CTAD**

TIPO: Contador $= \mathbb{N}$
INICIAL: *self* $= 0$
INVARIANTE: $-1 \leq$ *self* $\wedge$ *self* ≤ 1

CPRE: *self* < 1
inc()
POST: *self* $=$ *self*$^{pre} + 1$
CPRE: *Cierto*
dec()
POST: *self* $= -1$

Se ha decidido implementarlo con monitores mediante el siguiente código:

```
public class contador{
  private Monitor mutex =
    new Monitor();
  private Monitor.Cond cond =
    mutex.newCond();
  private int valor = 0;

  public  void dec(){
    mutex.enter();
    this.valor = - 1;
    cond.signal();
    mutex.leave();}
```

```
  public void inc(){
    mutex.enter();
    if(this.valor >= 1)
      cond.await();
    this.valor ++;
    mutex.leave();}
}
```

Se pide marcar la afirmación correcta:

(a) Se trata de una implementación correcta del recurso.

(b) Podría llegar a violarse la invariante.

(c) Podría darse el caso de que hubiese hilos esperando en cond que, pudiendo ejecutarse, no se desbloqueen.

Apellidos: **Nombre:** **Matrícula:**

(1½ puntos) 5. ¿Cual de los siguientes códigos sería **una buena implementación** del método run() para este recurso si quisiéramos implementarlo en JCSP?

A

```
public void run()
{
  final int INC = 0;
  final int DEC = 1;
  int valor = 0;
  Guard[] ent =
    {petInc.in(),
     petDec.in()};
  Alternative ser =
    new Alternative (ent);
  boolean[] sc =
    new boolean[2];
  while(true){
    sc[INC] = valor < 1;
    sc[DEC] = true;
    int sel =
      ser.fairSelect(sc);
    ent[sel].read();
    switch (sel) {
      case INC:
        valor++;
        break;
      case DEC:
        valor = -1;
        break;}
  }}
```

B

```
public void run()
{
  final int INC = 0;
  final int DEC = 1;
  int valor = 0;
  Guard[] ent =
    {petInc.in(),
     petDec.in()};
  Alternative ser =
    new Alternative (ent);
  while(true){
    int sel =
      ser.fairSelect();
    ent[sel].read();
    switch (sel) {
      case INC:
        valor++;
        break;
      case DEC:
        valor = -1;
        break;}
  }}
```

C

```
public void run()
{
  final int INC = 0;
  final int DEC = 1;
  int valor = 0;
  Guard[] ent =
    {petInc.in(),
     petDec.in()};
  Alternative ser =
    new Alternative (ent);
  boolean[] sc =
    new boolean[2];
  while(true){
    sc[INC] = true;
    sc[DEC] = valor < 1;
    int sel =
      ser.fairSelect(sc);
    ent[sel].read();
    switch (sel) {
      case INC:
        valor++;
        break;
      case DEC:
        valor = -1;
        break;}
  }}
```

Se pide marcar la afirmación correcta:

(a) C.

(b) A.

(c) B.

(1 punto) 6. Supóngase que una condición de sincronización (*CPRE*) de una operación *Op* de un recurso compartido depende únicamente 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) Para implementar la sincronización condicional de *Op* es necesario crear una variable `Cond` por cada posible valor de x.

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

(3 puntos) 7. El siguiente recurso compartido forma parte de un algoritmo paralelo de ordenación por mezcla. Permite mezclar dos secuencias ordenadas de números enteros para formar una única secuencia ordenada. En este recurso interactúan **solo tres procesos**: dos productores (izquierdo y derecho) que van pasando números de sus secuencias de uno en uno y un consumidor que va extrayendo los números en orden.

C-TAD OrdMezcla
OPERACIONES
ACCIÓN insertar: *Lado[e]* $\times$ $\mathbb{Z}$*[e]*
ACCIÓN extraerMenor: $\mathbb{Z}$*[s]*
SEMÁNTICA
DOMINIO:
TIPO: $OrdMezcla = \langle hayDato : Lado \to \mathbb{B} \times dato : Lado \to \mathbb{Z} \rangle$
TIPO: $Lado = Izda \mid Dcha$
INICIAL: $\forall\, l \in Lado \bullet \neg self.hayDato(l)$

CPRE: $\neg$self.$hayDato(l)$
insertar(l,d)
POST: self$^{pre} = \langle hay, dat \rangle \wedge$ self $= \langle hay \oplus \{l \mapsto \text{Cierto}\}, dat \oplus \{l \mapsto d\} \rangle$

CPRE: $self.hayDato(Izda) \wedge self.hayDato(Dcha)$
extraerMenor(min)
POST: self$^{pre} = \langle hay, dat \rangle \wedge$
$(dat(Izda) \leq dat(Dcha) \Rightarrow min = dat(Izda) \wedge \text{self} = \langle hay \oplus \{Izda \mapsto \text{Falso}\}, dat \rangle) \wedge$
$(dat(Dcha) \leq dat(Izda) \Rightarrow min = dat(Dcha) \wedge \text{self} = \langle hay \oplus \{Dcha \mapsto \text{Falso}\}, dat \rangle)$

La operación *insertar(lado, dato)* inserta *dato* en el *lado* correspondiente, bloqueando si ese hueco no está disponible. Cuando hay datos de ambas secuencias la operación *extraerMenor* tomará el menor de ambos y permitirá que se añada un nuevo dato de la secuencia correspondiente. Por concisión, no hemos considerado el problema de la terminación de las secuencias.

Se pide: Completar la implementación de este recurso mediante monitores que aparece a continuación y en la página siguiente. En cuanto al código de desbloqueos podéis optar tanto por un método de desbloqueo genérico como por tener código de desbloqueo especializado en los métodos `insertar` y `extraerMenor`. Si optáis por la segunda posibilidad dejad en blanco el cuerpo del método `desbloqueoSimple`.

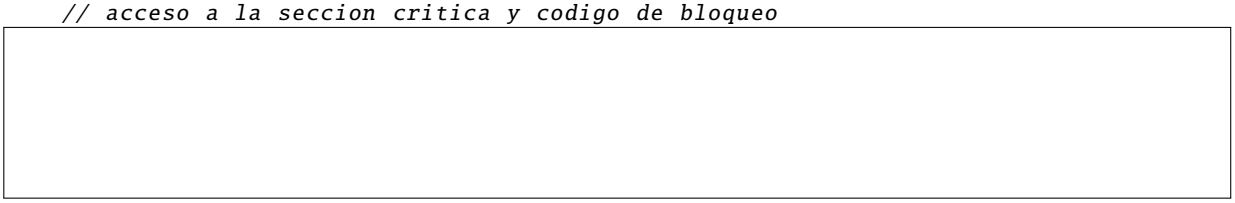
```
class OrdMezclaMon implements OrdMezcla {
  static final int IZDA = 0;
  static final int DCHA = 1;
  // estado del recurso
```

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

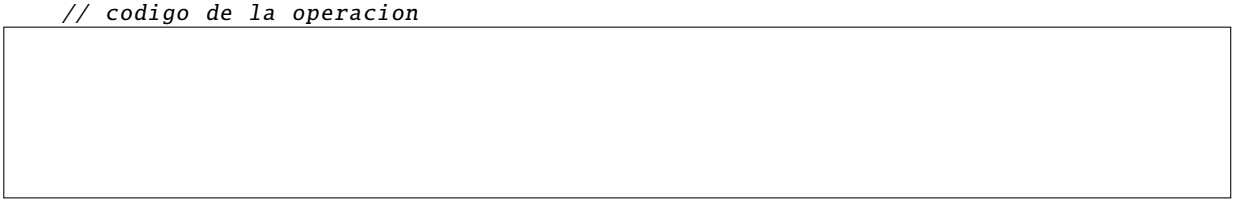
```
  public OrdMezclaMon() {
```

```
  }
```

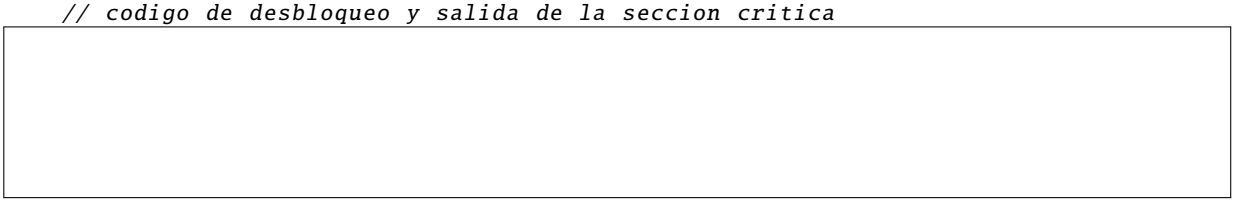
```
  public void insertar(int lado, int dato) {
    // acceso a la seccion critica y codigo de bloqueo
```



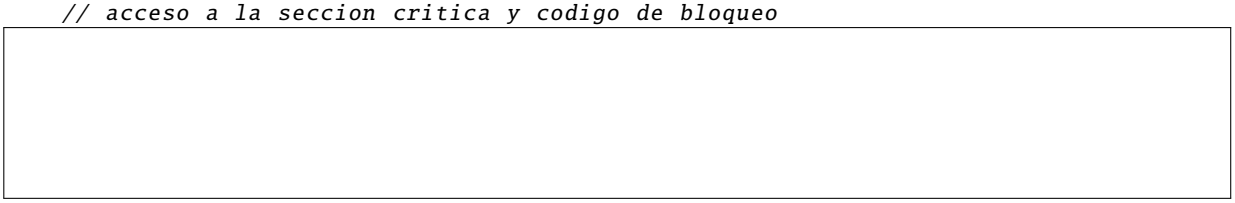
```
    // codigo de la operacion
```



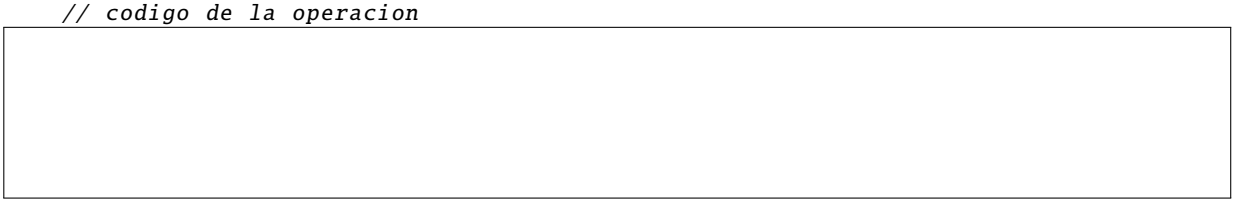
```
    // codigo de desbloqueo y salida de la seccion critica
```



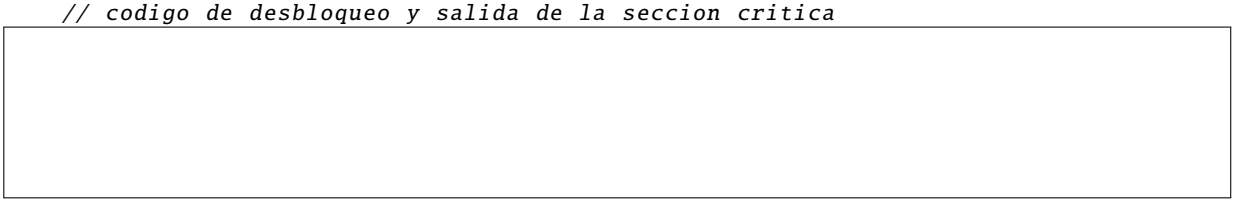
```
  }

  public int extraerMenor() {
    int result;
    // acceso a la seccion critica y codigo de bloqueo
```



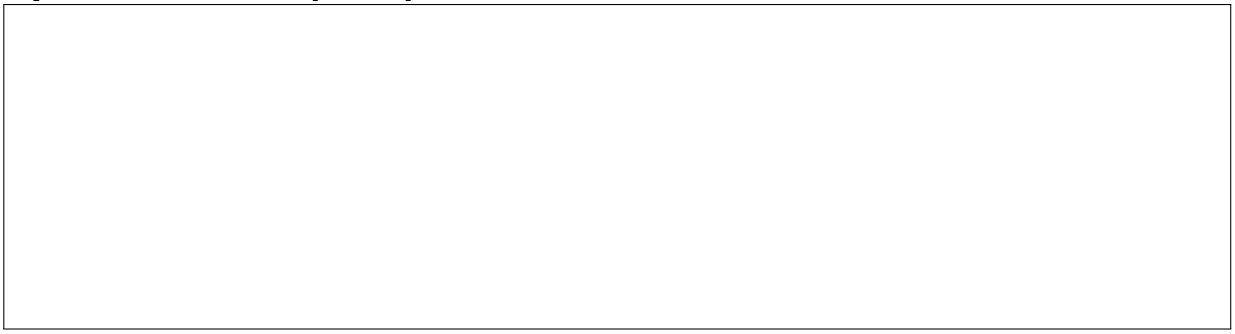
```
    // codigo de la operacion
```



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



```
  return result;
  }

  private void desbloqueoSimple() {
```



```
  }
}
```