Apellidos: **Nombre:** **Matrícula:**

Prueba objetiva 1 - Clave a

Concurrencia

2010-2011 - Primer semestre

Lenguajes, Sistemas Informáticos e Ingeniería de Software

Normas

Este es un cuestionario que consta de **5 preguntas de respuesta simple** y **una pregunta de desarrollo** en **6 páginas**. 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 DNI** 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.

Cuestionario

(2 puntos) 1. Dado el siguiente **CTAD** (sólo se muestran las partes necesarias):

TIPO: T_Cuatro $= (a : \mathbb{B} \times b : \mathbb{B})$

INICIAL(r): $r = (cierto, cierto)$

CPRE: *cierto*
S(r)
POST: $r^{pre} = (pe, se) \wedge r = (ps, \neg se) \wedge ps = (\neg pe \wedge se) \vee (pe \wedge \neg se)$

CPRE: $r \neq (cierto, falso)$
M(r)
POST: $r = (\neg r^{pre}.a, r^{pre}.b)$

Se pide marcar la afirmación correcta:

(a) El recurso puede pasar, a lo sumo, por 3 estados diferentes.

(b) El recurso podría estar en un estado dado y, tras ejecutarse una sola de sus acciones, continuar en el mismo estado.

(c) Si el sistema solo contiene (además de un recurso de este tipo) un único proceso que intenta ejecutar *S* indefinidamente, el sistema no puede acabar en interbloqueo.

(d) Si el sistema consta de un recurso de este tipo, un proceso que invoca a *S* una sola vez y otro que intenta ejecutar *M* indefinidamente, el sistema no puede terminar en interbloqueo.

PISTA: Dibujar el grafo de estados en este hueco:

(2 puntos) 2. Dado el siguiente **CTAD** (sólo se muestran las partes necesarias):

TIPO: Factoría $= \mathit{Área} \nrightarrow \mathbb{N}$
DONDE: $\mathit{Área} = \{0, 1\}$
INVARIANTE: $\forall f \in \mathit{Factoría}.\ (\forall\, a \in \mathit{Área}.\, f(a) \leq 10) \wedge \sum_{a \in \mathit{Área}} f(a) < 20$

INICIAL(f): $\sum_{a \in \mathit{Área}} f(a) = 0$

CPRE: $f(a) < 10 \wedge \sum_{a \in \mathit{Área}} f(a) < 19$
Entrar(f,a)
POST: $f = f^{pre} \oplus \{a \mapsto f^{pre}(a) + 1\}$

CPRE: Cierto
Salir(f,a)
POST: $f = f^{pre} \oplus \{a \mapsto f^{pre}(a) - 1\}$

Supóngase un programa concurrente en el que los procesos respetan el siguiente protocolo (o esquemas de llamada): $\mathit{Entrar}(f, a);\ \ldots;\ \mathit{Salir}(f, a)$.

Se pide señalar la respuesta correcta (asumir, obviamente, que la invariante se cumple antes de la invocación de cada operación).

(a) El programa puede violar la invariante del recurso compartido sólo en la operación *Entrar*.

(b) El programa puede violar la invariante del recurso compartido sólo en la operación *Salir*.

(c) El programa puede violar la invariante del recurso compartido tanto en *Entrar* como en *Salir*.

(d) Ninguna de las otras respuestas es correcta.

(1 punto) 3. La clase `PorTurno` implementa un protocolo de acceso a una sección crítica:

```
static int turno = 0;

static class PorTurno extends Thread {
    private int pid;

    public PorTurno(int pid) {
        this.pid = pid;
    }
```

```
    public void run() {
        while (true) {
            s();
            while (turno != pid) {}
            seccionCritica();
            turno = (turno + 1)
                    % MAX_THREADS;
        }
    }
}
```

Dado un programa concurrente con `MAX_THREADS` *threads* de la clase `PorTurno` compartiendo una variable `turno` inicializada a `0` y cada una de ellas con un índice `pid` distinto entre `0` y `MAX_THREADS-1`.

Se pide marcar la afirmación correcta.

(a) El programa no cumple la propiedad de exclusión mutua en `seccionCritica()`.

(b) Garantizada la terminación de `s()` y `seccionCritica()`, el programa no cumple la propiedad de ausencia de interbloqueo.

(c) Garantizada la terminación de `s()` y `seccionCritica()`, el programa no cumple la propiedad de ausencia de inanición.

(d) Garantizada la terminación de `s()` y `seccionCritica()` el programa cumple las propiedades de exclusión mutua en `seccionCriticar()`, ausencia de interbloqueo y ausencia de inanición pero un proceso podría esperar para ejecutar `seccionCritica()` sin que los demás ejecuten `seccionCritica()` ni compitan para hacerlo.

Apellidos: **Nombre:** **Matrícula:**

(1 punto) 4. Dado un programa concurrente en la que tres *threads* instancias de las clases `C`, `D` y `E` comparten una variable `n`:

```
static int n = 0;
static Semaphore s1 =
    new Semaphore(1);
static Semaphore s2 =
    new Semaphore(0);

static class C extends Thread {
    public void run() {
        s2.acquire();
        s1.acquire();
        n = 2 * n;
        s1.release();
    }
}
```

```
static class D extends Thread {
    public void run() {
        s1.acquire();
        n = n * n;
        s1.release();
    }
}

static class E extends Thread {
    public void run() {
        s1.acquire();
        n = n + 3;
        s2.release();
        s1.release();
    }
}
```

Se pide marcar el conjunto que contiene únicamente los posibles valores de la variable `n` tras la terminación de los tres threads.

(a) $\{3, 6, 9, 18, 36\}$

(b) $\{3, 6, 18, 36\}$

(c) $\{6, 18, 36\}$

(d) No está garantizada la terminación de las tres tareas.

(1 punto) 5. La instrucción TST (*Test and set*) es típica de algunas arquitecturas. Su comportamiento se basa en la existencia de una variable `c` común a varios procesos. Al ejecutar `x = TST();`, donde `x` debería ser una variable local al proceso, se puede asumir que se realiza **atómicamente** la siguiente ejecución: `x = c; c = 1`. El siguiente programa concurrente hace uso de dicha instrucción para regular el acceso a una sección crítica:

```
public static final
    void main(final String[] args) {
  Thread t1, t2;
  t1 = new T(); t2 = new T();
  t1.start(); t2.start();
}
```

```
static class T extends Thread {
  public T () { }
  public void run() {
    int x;
    while (true) {
      Sec_No_Critica();
      do x = TST(); while (x != 0);
      Sec_Critica();
      c = 0;
    }
  }
}
```

Se pide marcar la afirmación correcta:

(a) No se garantiza la propiedad de exclusión mutua.

(b) Se puede producir interbloqueo.

(c) Puede producirse inanición de un proceso.

(d) Se garantiza la exclusión mutua y la ausencia de inanición sin que haya posibilidad de interbloqueo.

(3 puntos) 6. Un puente sólo permite el paso de vehículos o bien en sentido N-S o en sentido S-N, pero no simultáneamente. Para controlar el paso de los coches se instalan, en ambas entradas – N y S – un detector de coches y una barrera (del mismo fabricante que en el ejemplo del parking), con operaciones *DetectarCoche* y *AbrirBarrera*. Asímismo, se instalan en ambas salidas sendos detectores.

Se pide especificar un recurso compartido *Puente* que controle el acceso al puente con la siguiente interfaz:

C-TAD Puente

OPERACIONES

ACCIÓN NotificarLlegada: *TipoPuente [es]* × *TipoSentido [e])*
ACCIÓN SolicitarEntrada: *TipoPuente [es]* × *TipoSentido [e])*
ACCIÓN NotificarSalida: *TipoPuente [es]* × *TipoSentido [e])*

SEMÁNTICA
DOMINIO:
TIPO: *TipoPuente* = ...
TIPO: *TipoSentido* = *N-S* | *S-N*

[...]

El recurso debe garantizar que los coches acceden al puente sin chocar con los que vienen en sentido contrario (seguridad) y que todo coche que llega a una de las entradas acaba cruzando el puente (ausencia de inanición).

A modo de información adicional, supondremos que los threads que implementan el sistema ejecutan, **en un bucle**, los siguientes esquemas de llamadas:

```
ControlEntradaS:
  DetectarCoche(1);
  NotificarLlegada(puente, S);
  SolicitarEntrada(puente, S);
  AbrirBarrera(1);

ControlEntradaN:
  DetectarCoche(2);
  NotificarLlegada(puente, N);
  SolicitarEntrada(puente, N);
  AbrirBarrera(2);

ControlSalidaN:
  DetectarCoche(3);
  NotificarSalida(N);

ControlSalidaS:
  DetectarCoche(4);
  NotificarSalida(S);
```

Apellidos: **Nombre:** **Matrícula:**

(Escribe aquí la solución a la pregunta de desarrollo.)

(Página intencionadamente en blanco.)