Prueba objetiva final - Clave a

Concurrencia

2009-2010 - Segundo semestre

Lenguajes y Sistemas Informáticos e Ingeniería de Software

Normas

Este es un cuestionario tipo test que consta de **5 preguntas** en **2 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 por pregunta. 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: Factoría $= \mathit{Área} \nrightarrow \mathbb{N}$
DONDE: $\mathit{Área} = \{0, 1\}$
INVARIANTE: $\forall f \in \mathit{Factoría}.\ (\forall\, a \in \mathit{Área}.\, f(a) \leq 2) \wedge \sum_{a \in \mathit{Área}} f(a) < 4$

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

CPRE: $f(a) < 2 \wedge \sum_{a \in \mathit{Área}} f(a) < 4$
Entrar(f,a)
POST: $f^{sal} = f^{ent} \oplus \{a \mapsto f^{ent}(a) + 1\}$

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

Supóngase un programa concurrente en el que los procesos respetan el siguiente protocolo (o esquemas de llamada): *Entrar*(f, a); ...; *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.

(2 puntos) 2. Asumiendo que en el recurso compartido del multibuffer ningún proceso realiza inserciones o extracciones de más datos que la mitad de la capacidad del multibuffer.

Se pide señalar la respuesta correcta:

(a) Nunca hay procesos bloqueados.

(b) Siempre hay procesos bloqueados.

(c) Nunca hay procesos de los dos tipos bloqueados.

(d) Ninguna de las otras respuestas es correcta.

(2 puntos) 3. A continuación se muestra una implementación del recurso de la pregunta 1 utilizando los métodos *synchronized*, **wait** y **notify** de Java:

```
static class Factoria {
  private int dentro[] = {0, 0};
  private int total = 0;

  public Factoria() { }

  public synchronized void
    salir(int a) {
      dentro[a]--; total--;
      notify();
    }
```

```
  public synchronized void
    entrar(int a) {
      if (dentro[a]==2 || total==4) {
        try {
          wait();
        }
        catch (Exception e) { }
      }
      dentro[a]++; total++;
    }
}
```

Se pide señalar la respuesta correcta:

(a) Es una implementación correcta del recurso.

(b) Provoca interbloqueos.

(c) Provoca inanición.

(d) Provoca la ejecución de una operación cuando su condición de sincronización (*CPRE*) es falsa.

(2 puntos) 4. A continuación se muestran las partes relevantes (cliente y servidor) de una implementación del recurso del problema 1 utilizando paso de mensajes.

```
private Any2OneChannel chEntrar = Channel.any2one();
```

```
// Ejecutado por el cliente
public void entrar(int a) {
  One2OneChannel sincro =
    Channel.one2one();
  Object[] pet =
    {new Integer(a),sincro};
  chEntrar.out().write(pet);
  sincro.out().write(null);
}
```

```
// Entrada en la select del servidor
case ENTRAR:
  Object[] pet =
    (Object[])chEntrar.in().read();
  int a = ((Integer)pet[0]).intValue();
  if (dentro[a] < 2 && total < 4) {
    dentro[a]++; total++;
    ((One2OneChannel)pet[1]).in().read();
  }
  else
    esperanEntrar.enqueue(pet);
  break;
```

Como se puede observar se ha modificado el esquema metodológico seguido en clase con *send-receive* en el cliente y *receive-send* en el servidor por esquemas *send-send* y *receive-receive* respectivamente.

Se pide señalar la respuesta correcta.

(a) Con dicho cambio el código no compilaría.

(b) Para este ejemplo el esquema propuesto es igualmente válido.

(c) Para este ejemplo el esquema propuesto es incorrecto.

(d) Ninguna de las otras respuestas es correcta.

(2 puntos) 5. 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). Supóngase que dicho recurso va a ser implementado con *Locks* y *Conditions* 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 `Condition` por cada posible valor de x.

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

(c) El problema hay que implementarlo con métodos *synchronized* de Java.

(d) Ninguna de las otras respuestas es correcta.