

## Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software

Este examen consta de tres apartados: un planteamiento de un problema con una solución incompleta, un cuestionario y una pregunta de implementación. La duración total es de **tres horas**. El cuestionario debe contestarse **en la misma hoja** en que se os entrega (consultad con un profesor si necesitáis otra hoja) y se podrá devolver hasta **una hora** después de recibirlo. Al hacerlo se os dará la pregunta de implementación.

La puntuación total del examen es de **10 puntos**. Las soluciones se proporcionarán antes de la revisión. Las calificaciones se darán a conocer el día **15 de septiembre**. La revisión del examen tendrá lugar el día **19 de septiembre**.

## 1. Tabla con acceso concurrente

Se dispone de una estructura de datos que ofrece una funcionalidad similar a la de una tabla con las operaciones:

**Insertar:** inserta un dato en un determinado registro (clave).

**Buscar:** dado un dato busca el registro en el que está almacenado.

Dicha estructura permite la ejecución simultánea de varias operaciones de inserción siempre y cuando éstas no se realicen sobre el mismo registro. Cuando se está ejecutando una operación de búsqueda no es seguro realizar ninguna otra operación. Para evitar entrelazados indeseados se ha encapsulado una de las mencionadas estructuras de datos (T) junto con un recurso compartido (CA) que realiza la función de gestor de sincronización quedando las operaciones implementadas de la siguiente forma:

|  |   |
|--|---|
| <pre> <b>procedure</b> Insertar (D : <b>in</b> Dato;                     R : <b>in</b> Registro) <b>is</b>   <b>begin</b>     Iniciar_Insercion (CA,R);     Insertar (T,R,D);     Finalizar_Insercion (CA,R);   <b>end</b> Insertar;</pre> | <pre> <b>procedure</b> Buscar (D : <b>in</b> Dato;                   R : <b>out</b> Registro) <b>is</b>   <b>begin</b>     Iniciar_Busqueda (CA);     Buscar (T,D,R);     Finalizar_Busqueda (CA);   <b>end</b> Buscar;</pre> |
|--|---|

La especificación del recurso compartido *Control.Acceso* (CA es uno de estos recursos) es la siguiente:

**C-TADSOL** Control\_Acceso

### OPERACIONES

**ACCIÓN** Iniciar\_Insercion:  $\text{Control\_Acceso}[es] \times \text{Registro}[e]$

**ACCIÓN** Finalizar\_Insercion:  $\text{Control\_Acceso}[es] \times \text{Registro}[e]$

**ACCIÓN** Iniciar\_Busqueda:  $\text{Control\_Acceso}[es]$

**ACCIÓN** Finalizar\_Busqueda:  $\text{Control\_Acceso}[es]$

### SEMÁNTICA

#### DOMINIO:

**TIPO:**  $\text{Control\_Acceso} = (\text{busqueda}: \mathbb{B} \times \text{insercion}: \text{Conjunto}(\text{Registro}))$

**INVARIANTE:**  $\forall ca \in \text{Control\_Acceso} \bullet \neg(ca.\text{busqueda} \wedge \text{Card}(ca.\text{insercion}) > 0)$

**INICIAL(r):**  $ca = (\text{Falso}, \emptyset)$

**CPRE:**  $r \notin ca.\text{insercion} \wedge \neg ca.\text{busqueda}$

**Iniciar\_Insercion(ca,r)**

**POST:**  $ca^{sal}.\text{busqueda} = ca^{ent}.\text{busqueda} \wedge$   
 $ca^{sal}.\text{insercion} = ca^{ent}.\text{insercion} \cup \{r\}$

**CPRE:** Cierto

**Finalizar\_Insercion(ca,r)**

**POST:**  $ca^{sal}.\text{busqueda} = ca^{ent}.\text{busqueda} \wedge$   
 $ca^{sal}.\text{insercion} = ca^{ent}.\text{insercion} - \{r\}$

**CPRE:**  $ca.\text{insercion} = \emptyset \wedge \neg ca.\text{busqueda}$

**Iniciar\_Busqueda(ca)**

**POST:**  $ca^{sal}.\text{busqueda} \wedge$   
 $ca^{sal}.\text{insercion} = ca^{ent}.\text{insercion}$

**CPRE:** Cierto

**Finalizar\_Busqueda(ca)**

**POST:**  $\neg ca^{sal}.\text{busqueda} \wedge$   
 $ca^{sal}.\text{insercion} = ca^{ent}.\text{insercion}$



Apellidos: \_\_\_\_\_

Nombre: \_\_\_\_\_ Número de matrícula: \_\_\_\_\_

## 2. Cuestionario

[5 puntos]

**Importante:** Cada cuestión de esta parte está valorada en **1 punto**. Para cada pregunta marcad la respuesta que consideréis correcta. 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 **0.25 puntos** de la puntuación total del examen.

### 2.1. Interbloqueos

Dos tipos de procesos necesitan acceder a recursos compartidos (por ejemplo, impresoras o discos), que tras ser utilizados se devuelven intactos. Los procesos de tipo P1 sólo necesitan una unidad del recurso; los recursos del tipo P2 necesitan dos unidades. Se plantea arbitrar el acceso al recurso mediante un único semáforo general Cont inicializado con la cantidad de recurso inicialmente disponible. Los procesos tienen el siguiente código:

```
P1:
loop
  Wait(Cont);
  <Usa una unidad del recurso
  y la devuelve al terminar>
  Signal(Cont);
end loop;
```

```
P2:
loop
  Wait(Cont);
  Wait(Cont);
  <Usa dos unidades del recurso
  y las devuelve al terminar>
  Signal(Cont);
  Signal(Cont);
end loop;
```

Asumimos que Cont se inicializa con el valor 2. No podemos hacer ninguna suposición sobre la planificación de los procesos: cualquier entrelazado legal es posible, y no se asegura que ninguno en particular vaya a producirse. **Se pide:** marcar cuál de las siguientes afirmaciones es verdadera.

- ☐ Si hay más de un proceso de tipo P2 puede haber interbloqueo, sin importar cuantos haya de tipo P1.
- ☐ Nunca puede haber interbloqueos (hay el mismo número de **Signal** que de **Wait**).
- ☐ Puede haber interbloqueos con cualquier número de procesos de cualquier tipo.
- ☐ Es posible que haya interbloqueos si hay sólo procesos de tipo P1.

### 2.2. Seguridad

Supongamos el mismo planteamiento que en la pregunta 2.1. **Se pide:** marcar cuál de las siguientes afirmaciones es verdadera.

- ☐ Si sólo hay procesos de tipo P2, es posible que haya accesos al recurso que no se realizan en exclusión mutua entre procesos.
- ☐ Si hay un proceso de tipo P1 y otro de tipo P2 es posible que haya accesos al recurso que no se realizan en exclusión mutua entre procesos.
- ☐ Si sólo hay procesos de tipo P1, es posible que haya accesos al recurso que no se realizan en exclusión mutua entre procesos.
- ☐ Si hay un número indeterminado de procesos de tipo P1 y de tipo P2 es seguro que, en algún momento, habrá accesos al recurso que no se realizan en exclusión mutua entre procesos.

### 2.3. Invariante

Tenemos un sistema basado en paso de mensajes síncrono cuya ejecución no termina nunca y del que sabemos que ningún proceso se queda nunca indefinidamente bloqueado. **Se pide:** decidir cuál de las siguientes afirmaciones es correcta.

- ☐ Ningún proceso queda suspendido al intentar enviar un mensaje.
- ☐ El número de mensajes enviados es siempre mayor o igual que el de mensajes recibidos.
- ☐ Ningún proceso queda suspendido al intentar recibir un mensaje.
- ☐ Pueden recibirse mensajes repetidos.

### 2.4. Vivacidad

La especificación del apartado 1 podría tener un problema: sospechamos que es posible que, aunque haya una planificación equitativa que dé oportunidades a todos los procesos por igual, un tipo de procesos monopolice la estructura de datos e impida el acceso a procesos del otro tipo. **Se pide:** de entre las siguientes opciones elegir la correcta.

- ☐ No hay ningún problema de vivacidad. Si la planificación es equitativa todos los procesos que quieran acceder a la tabla acabarán haciéndolo más tarde o más temprano.
- ☐ Los procesos que quieren insertar (**PI**) pueden impedir el acceso a los que quieren buscar (**PB**) *ad infinitum*.
- ☐ Los procesos **PB** pueden impedir el acceso a los **PI** *ad infinitum*.
- ☐ Es peor de lo esperado: existe la posibilidad de que el sistema se paralice con todos los procesos de tipo **PB** y todos los de tipo **PI** bloqueados.

### 2.5. Resultado final

Tenemos un tipo protegido (mostrado a la izquierda) y tres procesos P1, P2 y P3 (mostrados a la derecha). Los procesos se ejecutan en un orden desconocido.

```

protected type T is
  entry A;
  entry B;
  entry C;
private
  M : Integer := 0;
end T;

protected body T is
  entry A when M >= 0 is
    begin
      M := M - 1;
    end A;
  entry B when M <= 0 is
    begin
      M := M + 1;
    end B;
  entry C when M = 0 is
    begin
      M := 3;
    end C;
end T;

```

```

Comp: T;

P1:
  begin
    Comp.A;
  end;

P2:
  begin
    Comp.B;
  end;

P3:
  begin
    Comp.C;
  end;

```

**Se pide:** determinar, de entre las posibilidades que aparecen más abajo, el valor/valores que puede tomar la variable de estado M cuando los tres procesos hayan acabado su ejecución.

- ☐ 1.
- ☐ 2.
- ☐ 3.
- ☐ Debido al no determinismo de la ejecución no hay un único valor en el que pueda estar M al terminar la ejecución de los procesos P1 a P3.

## Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software

Este examen consta de tres apartados: un planteamiento de un problema con una solución incompleta, un cuestionario y una pregunta de implementación. La duración total es de **tres horas**. El cuestionario debe contestarse **en la misma hoja** en que se os entrega (consultad con un profesor si necesitáis otra hoja) y se podrá devolver hasta **una hora** después de recibirlo. Al hacerlo se os dará la pregunta de implementación.

La puntuación total del examen es de **10 puntos**. Las soluciones se proporcionarán antes de la revisión. Las calificaciones se darán a conocer el día **15 de septiembre**. La revisión del examen tendrá lugar el día **19 de septiembre**.

### 3. Implementación

[5 puntos]

Para cada uno de los apartados siguientes ha de entregarse lo que se pide en ellos. Recordad **entregar la respuesta a cada apartado en un juego de hojas separadas**.

#### 3.1. Objetos protegidos

[2.5 puntos]

Realizar la implementación de todas las operaciones del recurso compartido `ControlAcceso` utilizando objetos protegidos como mecanismo de concurrencia, aplicando la metodología propia de la asignatura y siguiendo el esquema que se ofrece en el apartado 3.3.

#### 3.2. *Rendez-Vous* / Paso de mensajes

[2.5 puntos]

Completar la implementación de todas las operaciones del recurso compartido `ControlAcceso` utilizando *rendez-vous* y paso de mensajes como mecanismos de concurrencia, aplicando la metodología propia de la asignatura y siguiendo el esquema que se ofrece en el apartado 3.3.

#### 3.3. Esquema para completar la implementación

`control.accesos.ads` (no hay que modificarlo o ampliarlo, sólo respetarlo)

— Importación del módulo con el tipo *PRIVADO Registro*

```
with ...;
use ...;

package ControlAccesos is
  type ControlAcceso is private;

  procedure Iniciar_Insercion (CA : in out ControlAcceso; R : in Registro);
  procedure Finalizar_Insercion (CA : in out ControlAcceso; R : in Registro);
  procedure Iniciar_Busqueda (CA : in out ControlAcceso);
  procedure Finalizar_Busqueda (CA : in out ControlAcceso);

private
  type ControlAcceso_Impl;
  type ControlAcceso is access ControlAcceso_Impl;
end ControlAccesos;
```

**control\_accesos.adb (esquema)**


---

— *Importación de paquetes que sean necesarios: a completar*

---

**package body** Control.Accesos **is**

---

— *Tipos auxiliares e instanciación de paquetes genéricos: a completar*

---

— *Declaración del tipo como objeto protegido o como servidor*  
**protected** ó **task type** Control.Acceso\_Impl **is**

— *A completar*  
**end** Control.Acceso\_Impl;

---

— *Implementación del tipo como objeto protegido o como servidor*  
**protected** ó **task body** Control.Acceso\_Impl **is**

— *A completar*  
**end** Control.Acceso\_Impl;

---

**function** Crear\_Control\_Acceso **return** Control\_Acceso **is**

— *A completar*  
**begin**  
 — *A completar*  
**end** Crear\_Control\_Acceso;

---

**procedure** Iniciar\_Insercion (CA : **in out** Control\_Acceso; R : **in** Registro)

— *A completar*  
**begin**  
 — *A completar*  
**end** Iniciar\_Insercion;

---

**procedure** Finalizar\_Insercion (CA : **in out** Control\_Acceso; R : **in** Registro)

— *A completar*  
**begin**  
 — *A completar*  
**end** Finalizar\_Insercion;

---

**procedure** Iniciar\_Busqueda (CA : **in out** Control\_Acceso)

— *A completar*  
**begin**  
 — *A completar*  
**end** Iniciar\_Busqueda;

---

**procedure** Finalizar\_Busqueda (CA : **in out** Control\_Acceso)

— *A completar*  
**begin**  
 — *A completar*  
**end** Finalizar\_Busqueda;  
**end** Control\_Accesos;