# Examen de Programación Concurrente y Distribuida

### 3º Curso de Grado en Ingeniería Informática

## Septiembre. Curso 2019-20

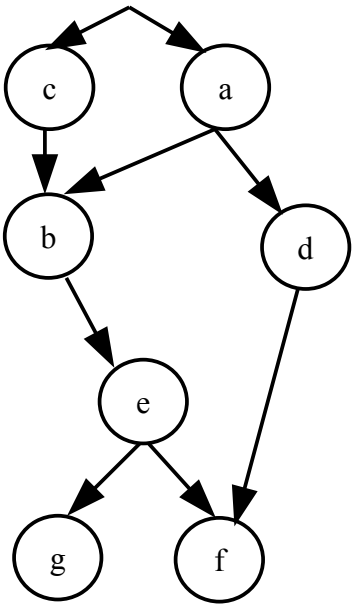
**ANTES DE COMENZAR:**

- **Apague el móvil y quítelo de encima del pupitre.**
- **Ponga su nombre en todos los folios que tenga.**
- **Cada pregunta debe responderse en un folio distinto.**

1. Explique brevemente qué problemas plantea el paso de parámetros en las llamadas a procedimiento remoto (RPC) y cuales son sus posibles soluciones. **(0,5 Puntos)**

2. Dado el siguiente programa, corríjalo, usando la menor cantidad posible de semáforos, para que se cumpla el grafo de precedencias que se indica. No olvide inicializar los semáforos que use. **(1,5 Punto)**

```
Program P
process P1
begin
    a;
    d;
end
process P2
begin
    c;
    e;
    g;
end
process P3
begin
    b;
    f;
end
begin
    conbegin
      P1; P2; P3;
    coend
end
```



3. Usando semáforos, haga que, de forma cíclica, los procesos accedan a la sección crítica en la siguiente secuencia: P1, P1, P2, P1, P1,P3 P1, P1, P2, P1, P1,P3 …. . **(1,5 Puntos)**

**No se considera válida la solución si no se inicializan los semáforos correctamente**

```
Program dtdtu
var
process P1
begin
 repeat
   Sección Crítica
   Resto1
 forever
end
```

```
process P2
begin
 repeat
   Sección Crítica
   Resto2
 forever
end
```

```
process P3
begin
 repeat
   Sección Crítica
   Resto3
 forever
end
```

```
begin
 cobegin
      P1;P2;P3;
 coend
end
```

4. Tenemos un sistema operativo con 5 procesos (P1, P2, P3, P4 y P5). En el sistema tenemos 4 recursos: 2 ejemplares de R1, 3 ejemplares de R2, 2 ejemplares de R3 y 3 ejemplares de R4. En un momento dado, el sistema se encuentra en la siguiente situación:

**Necesidades máximas**

| | **R1** | **R2** | **R3** | **R4** |
|---|---|---|---|---|
| **P1** | 1 | 2 | 0 | 0 |
| **P2** | 1 | 1 | 2 | 0 |
| **P3** | 0 | 1 | 0 | 2 |
| **P4** | 1 | 1 | 1 | 0 |
| **P5** | 0 | 0 | 1 | 1 |

**Recursos asignados**

| | **R1** | **R2** | **R3** | **R4** |
|---|---|---|---|---|
| **P1** | 0 | 2 | 0 | 0 |
| **P2** | 1 | 0 | 2 | 0 |
| **P3** | 0 | 0 | 0 | 2 |
| **P4** | 1 | 0 | 0 | 0 |
| **P5** | 0 | 0 | 0 | 0 |

Si estamos usando el algoritmo del banquero para evitar los interbloqueos. ¿Debería concederse a P2 un ejemplar del recurso de R2?. Justifique la respuesta.. **(1 Punto)**.

5. Un centro de salud tiene tres salas de atención a los pacientes. Los pacientes que llegan al centro pasarán a cualquiera de las salas que están libres. Una vez dentro, se evaluará si el paciente es COVID19 o no (se decidirá aleatoriamente, con una probabilidad del 30%). Si el paciente es COVID19, no se admitirán más pacientes en el resto de las salas hasta que dicho paciente haya sido evacuado (los que ya estaban en las salas de atención, terminarán de ser atendidos). El tiempo que un paciente "normal" permanece en la sala de espera es de 1 a 3 segundos. El tiempo que un paciente COVID19 tarda en ser evacuado es de 4 a 6 segundos.

   a) Solucionar el problema anterior usando **monitores**. Se asume una semántica de la operación `resume` tipo "desbloquear y espera urgente" (la habitual de *Pascal-FC*). **(3 Puntos)**.

   b) Solucionar el problema anterior usando **canales**. La solución debe ser correcta para un sistema distribuido, donde los procesos estén en máquinas distintas. **(2,5 Puntos).**

**NOTAS:**

- **Para simplificar el código se usarán llaves `{` y `}` en lugar de las instrucciones `begin` y `end` para marcar los bloques de código.**
- **Si fuese necesario que un procedimiento devuelva un valor se puede usar `return`.**

**ANEXO 1. Estructura de los procesos para el problema 5**

```
program Febrero15;
const
    np=20;

process type Paciente(id:integer);
begin

end;

var
    i: integer;
    PA: array[1..np] of Paciente;

begin
    cobegin
         for i := 1 to np1 do PA[i](i);
    coend
end.
```