Examen de Programación Concurrente y Distribuida
3º Curso de Grado en Ingeniería Informática
Febrero. Curso 2021-22

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. Indique como el algoritmo del matón realiza la elección del proceso coordinador para sincronizar sistemas distribuidos. **(0,5 Puntos)**
2. Justifique si el siguiente algoritmo para el control de la concurrencia cumple las condiciones requeridas. **(0,75 Puntos)**

Inicialmente, turno vale 0,c0 vale 0 y c1 vale 0.

```
process P0
  repeat
1. c0 := 1;
2. while c1=1 and turno=1 do;
3. Sección Crítica
4. c0 := 0;
5. turno:= 1;
6. Resto0
  forever
```

```
process P1
  repeat
1. c1 := 1;
2. while c0=1 and turno=0 do;
3. Sección Crítica
4. c1 := 0;
5. Turno:=0;
6. Resto1
  forever
```

Para referirse a una determinada secuencia de instrucciones, use los números de instrucción.

Falta de exclusión, si P1 entra en SC, luego puede entrar P0 también.

Se produce una falta de exclusión con la secuencia:

P1 (1); P1 (2); P1 (3); P2 (1); P2 (2); P2 (3)

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

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

```
Program uuudtt
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
```

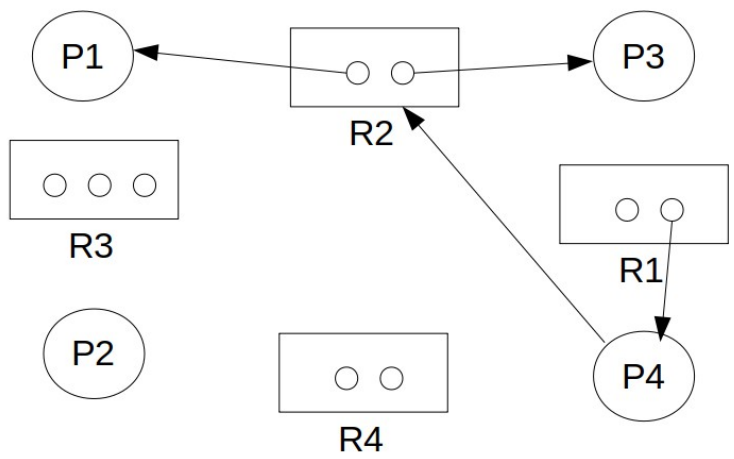
```
Program uuudtt
var
process P1
begin
 repeat
   wait(s1);
   Sección Crítica
   signal(s2);
   Resto1
 forever
end
```

```
process P2
begin
 repeat
   wait(s2)
   wait(s2)
   wait(s2)
   Sección Crítica
   signal(s3);
   signal(s3);
   Resto2
 forever
end
```

```
process P3
int v=0;
begin
 repeat
   wait(s3)
   Sección Crítica
  if(v=0) v++;
  else { v=0;
      signal(s1);
      signal(s1);
      signal(s1);
 }
  Resto3
 forever
end
```

```
begin
initial(s1,3);
initial(s2,0);
initial(s3,0);
 cobegin
      P1;P2;P3;
 coend
end
```

4. Tenemos un sistema operativo con 4 procesos, que en un momento dado presenta el siguiente estado:



y se sabe que las necesidades máximas de los procesos son:

	N. Máximas			
	R1	**R2**	**R3**	**R4**
P1	1	1	1	0
P2	0	0	2	1
P3	1	2	0	0
P4	1	1	0	1

Indique si se cumplen las condiciones para la existencia de un interbloqueo.

Las cuatro condiciones necesarias para que ocurra el interbloqueo son:

- Exclusión mutua
- No apropiación
- Retención y espera
- Espera circular.

En el sistema, las tres primeras son inherentes, por tanto se cumplen, pero **la cuarta no se produce al no haber ciclos en el grafo**. Por tanto, como todas son condiciones necesarias, podemos concluir que no se dan dichas condiciones.

Si estamos usando un algoritmo para evitar los interbloqueos, ¿debería concederse a P3 un ejemplar del recurso de R1?. Justifique la respuesta. **(1,25 Puntos)**.

- Tras P3 solicita R1 queda:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	1	0	0	1	1	1	0	1	0	1	0
P2	0	0	0	0	0	0	2	1	0	0	2	1
P3	1	1	0	0	1	2	0	0	0	1	0	0
P4	1	0	0	0	1	1	0	1	0	1	0	1

E= (2,2,3,2)
A= (2,2,0,0) -> L= (0,0,3,2)
y en esta situación podemos finalizar P2, pero luego no se puede localizar ninguna fila que sea menor o igual que L, con lo cual **el estado es inseguro, y por tanto esa solicitud no debería haberse concedido para evitar el interbloqueo.**
Tras finalizar P2 se comprueba que el nuevo vector de libres es L=(0, 0, 3 , 2) y que los vectores pendientes de P1, P2 y P4 son:
P4 -> (0,1,0,1) < L -> No
P3 -> (0,1,0,0) < L -> No
P1 -> (1,0,1,0) < L -> No
Con lo cual, el estado es inseguro.

5. En una estación de lavado de coches hay tres túneles de lavado dispuestos de forma paralela. A dicha estación llegan para lavarse turismos y furgonetas. Cualquiera de los vehículos podrá usar cualquiera de los túneles, pero si hay una furgoneta en el túnel central no podrá haber otra furgoneta en ninguno de los túneles laterales.

   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. **(3 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 Febrero22;
const
    nC=20;
    nF=20;

process type TCoche(id:integer);
begin

end;

process type TFurgo(id:integer);
begin

end;

var
    i,j: integer;
    Coche: array[1..nC] of TCoche;
    Furgo: array[1..nF] of TFurgo;

begin
    cobegin
         for i := 1 to nC do Coche[i](i);
         for j := 1 to nF do Furgo[i](i);
    coend
end.
```