Examen de Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

Septiembre. Curso 2014-15

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. Los multicomputadores conmutados están formados por multitud de nodos independientes, formados por procesador más memoria, e interconectados mediante alguna red de estructura regular. Represente gráficamente, tres de sus posibles topologías indicando su nombre correspondiente. **(0,5 Puntos)**

2. Dado el siguiente conjunto de instrucciones, utilice las condiciones de Bernstein para establecer el grafo de precedencias que le corresponde. **(0,75 Puntos)**

```
S1: a = c + b;
S2: d = c / b + f;
S3: c = x + 3;
S4: e = f * (c + 2);
S5: h = a * d;
S6: x = e / h;
```

3. Usando la instrucción hardware *addc*, garantice la exclusión mutua para los procesos P1 y P2. No olvide indicar el valor inicial de las variables que use. **(0,75 Puntos)**

```
process P1
repeat

   Sección Crítica

   Resto1
forever
```

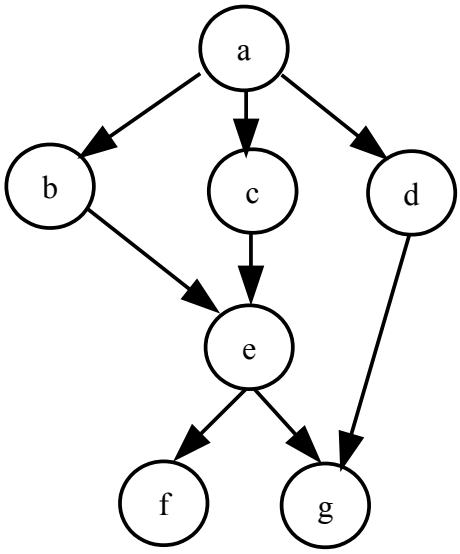
```
process P2
repeat

   Sección Crítica

   Resto2
forever
```

4. 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
```



5. 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)**.

6. Una linea de ensamblaje tiene tres puestos dispuestos linealmente etiquetados como T1, T2 y T3. A dicha linea de ensamblaje acuden los operarios con dos tipos distintos de productos llamados PA y PB. Los productos PA necesitan realizar los tratamientos T1 y T2 secuencialmente, pero los productos PB los hacen simultáneamente. El tratamiento T3 siempre se realiza el último. Los operarios con productos PB no podrán reservar los puestos T1 y T2 individualmente, sólo los podrán usar si están libres los dos.

   a) 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).**

   b) Solucionar el problema anterior usando **monitores**. En esta solución, los operarios con productos PB tendrán prioridad con respecto a los operarios con productos PA en el acceso a los puestos T1 y T2, pero sólo podrán usarlos si en el puesto T3 no hay otro operario con producto PB. Se asume una semántica de la operación `resume` tipo "desbloquear y espera urgente" (la habitual de *Pascal-FC*). **(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 6

```
program Febrero15;
const
    np1=10;
    np2=10;
process type OperarioA(id:integer);
begin

end;

process type OperarioB(id:integer);
begin

end;

var
    i,j: integer;
    PA: array[1..np1] of OperarioA;
    PB: array[1..np2] of OperarioB;

begin
    cobegin
         for i := 1 to np1 do PA[i](i);
         for j := 1 to np2 do PB[j](j);
    coend
end.
```