RELACIÓN DE PROBLEMAS 1

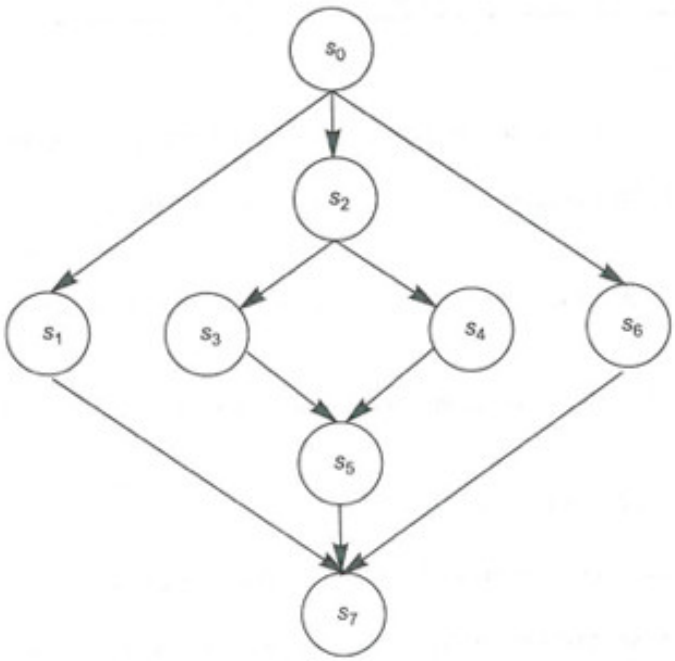
1.- Considerar las siguientes sentencias:

```
S1    a = 4;
S2    b = 6;
S3    c = a+5;
S4    d = b+7;
S5    e = 7;
S6    f = c*d;
S7    g = f/e;
```

Calcular los conjuntos de lectura, $L(S_k)$, y escritura, $E(S_k)$ utilizados en la definición de Berestein y determinár qué sentencias se pueden ejecutar en paralelo y cuales no. Además dibujar el grafo de precedencia asociado.

2.- Utilzar el par **cobegin-coend** que resuelve el grafo de precedencia obtenido en el ejercicio anterior.

3.- Dado el siguiente grafo de precedencia:

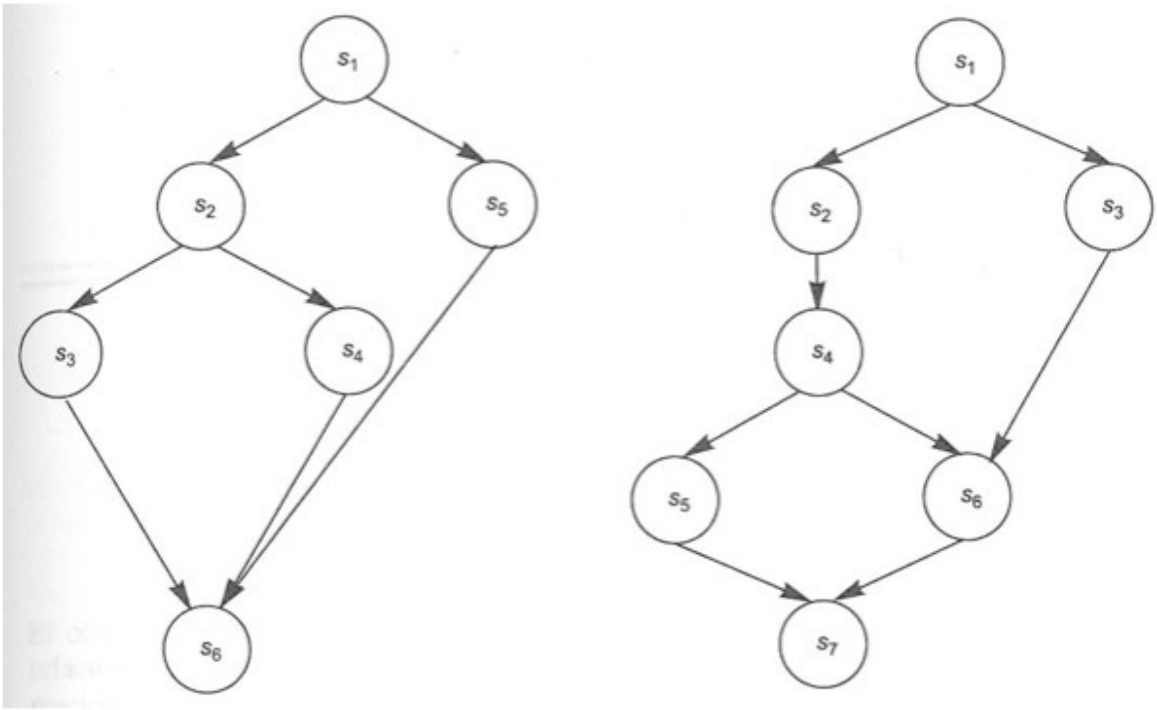


Utilizar el par **cobegin-coend** que resuelven el grafo anterior de precedencia.

4.- Usando las condiciones de Berstein, construir el grafo de precedencia del siguiente trozo de código y el programa concurrente correspondiente usando el par **cobegin-coend**.

```
S1 : cuad := x * x;
S2 : m1 := a * cuad;
S3 : m2 := b * x;
S4 : z := m1 + m2;
S5 : y := z + c;
```

5.- Construir dos programas concurrentes que se correspondan con los de la siguiente figura utilizando el par **cobegin-coend**.



6.- Considerar el siguiente fragmento de programa para dos procesos A y B:

Suponer que la variable x es una variable global que está inicializada a 0. ¿Cuáles son los posibles resultados para x?

```
process A;
var seqA:integer;
begin
   for seqA:=1 to 10 do x:=x+1
end;
```

```
process B;
var seqB:integer;
begin
   for seqB:=1 to 10 do x:=x+1
end;
```

6.- En el siguiente programa se espera que como resultado de su ejecución se imprima 110 ó 50. ¿Es correcto el algoritmo? Razone la respuesta.

```
program verbatim;
var x:integer;
process P1;
begin
   x:=x+10
end;
process P2;
begin
   if x>100
      then write(x)
      else write(x-50)
end;
begin
   x:=100;
   cobegin
      P1;P2;
   coend
end.
```

7.- Supongamos un sistema multiprocesador que posee una instrucción llamada TAS (TestAndSet). Ejecutando TAS(x) se obtendría lo mismo que con las dos sentencias siguientes: x:=c; c:=1;

Donde x es una variable local y c una variable global del sistema. Usando la instrucción anterior, construir los protocolos de entrada, de salida y la inicialización de las variables para dar una solución al problema de la exclusión mutua de las secciones críticas siguientes:

```
program testandset;
var c:integer;
process P1;
var x:integer;
begin
   repeat
      Resto1;
      Protocolo de Entrada;
      Sección Crítica1;
      Protocolo de Salida;
   forever
end;
process P2;
var x:integer;
begin
   repeat
      Resto2;
      Protocolo de Entrada;
      Sección Crítica2;
      Protocolo de Salida;
   forever
end;
begin
   inicialización;
   cobegin
      P1;P2;
   coend
end.
```

¿Qué ocurriría si en la solución anterior la instrucción TAS fuese remplazada por las dos sentencias a las que equivale? Razona la respuesta.

RELACIÓN DE PROBLEMAS 2

1.- Tenemos una BD que puede ser usada sólo por 5 procesos al mismo tiempo. Hay dos tipos de procesos: A y B. Modelice el acceso a la Base de Datos usando semáforos, teniendo en cuenta que:

- no puede haber más de 3 procesos A en la BD al mismo tiempo.
- no puede haber más de 4 procesos B en la BD al mismo tiempo.

2.- Se dispone de una sala de juegos y jugadores. Para entrar a la sala se necesitan exactamente 5 jugadores. A medida que los jugadores llegan esperan hasta completar 5; y luego entran a jugar a la sala. Para poder salir de la misma, todos deben haber terminado de jugar. Los jugadores pueden usar a lo sumo una variable compartida.

3.- En una sala con 30 filas de 10 asientos cada uno se realiza un congreso. La sala tiene dos entradas por donde entran los congresistas. Suponga que existen N congresistas que cuando llegan hacen cola en la entrada que **en ese momento tenga menos gente**. En caso que la sala esté llena el congresista se va.
Hay 3 acomodadores que van ubicando de a uno a los primeros de las colas hasta que la sala esté llena o hasta que termine la conferencia. Para acomodar a los congresistas los acomodadores buscan un asiento libre y lo dejan en la punta de la fila indicándoles su asiento, para que luego el congresista se siente.

4.- Los L procesos A se ejecutan utilizando la BD M y los K procesos B se ejecutan utilizando la BD N, cuando terminan de acceder a la BD todos los procesos usan la BD O luego de usarla salen de la ejecución.
Tenga en cuenta que no pueden acceder más de 4 procesos concurrentes a la BD M y no más de 5 a la BD N.
En la BD O siempre debe haber el mismo número de procesos A y B, la primera vez deben entrar cuatro procesos como mínimo, y una vez que fue accedida no puede quedar vacía.

5.- En una carpintería se realizan muebles ensamblando 3 partes, existen N1 carpinteros para realizar la parte 1, N2 para realizar la parte 2, N3 para la parte 3; mas N carpinteros que se encargan de ensamblar las 3 partes del mueble.
Los encargados de ensamblar deben tomar una pieza de cada clase juntar las partes y **una vez ensambladas** los carpinteros que le dieron la pieza pueden seguir trabajando (no antes). En la carpintería solo se montarán 30 muebles y **no** se podrán producir piezas de más.
El armado debe hacerse en forma **concurrente** sólo pueden esperarse entre los carpinteros de un mismo tipo cuando terminan una pieza hasta que los tome un ensamblador.

RELACIÓN DE PROBLEMAS 3

Monitores

1.- Se consideran dos recursos denominados r1 y r2. Del recurso r1 existen N1 ejemplares y del recurso r2 existen N2 ejemplares. Escribir un monitor que gestione la asignación de los recursos a los procesos de usuario, suponiendo que cada proceso puede pedir:

- un ejemplar del recurso r1
- un ejemplar del recurso r2
- un ejemplar del recurso r1 y otro del recurso r2

La solución deberá satisfacer estas dos condiciones:

- Un recurso no será asignado a un proceso que demande un ejemplar de r1 o un ejemplar de r2 hasta que al menos un ejemplar de dicho recurso quede libre.
- Se dará prioridad a los procesos que demanden un ejemplar de ambos recursos.

2.- Una cuenta de ahorros es compartida por varias personas (procesos). Cada persona puede depositar o retirar fondos de la cuenta. El saldo actual de la cuenta es la suma de todos los depósitos menos la suma de todos los reintegros. El saldo nunca puede ser negativo.

A. Programar un monitor para resolver el problema, todo proceso puede retirar fondos mientras la cantidad solicitada **c** sea menor o igual que el saldo disponible en la cuenta en ese momento. Si un proceso intenta retirar una cantidad **c** mayor que el saldo, debe quedar bloqueado hasta que el saldo se incremente lo suficiente (como consecuencia de que otros procesos depositen fondos en la cuenta) para que se pueda atender la petición. El monitor debe tener 2 procedimientos: **depositar(c)** y **retirar(c)**. Suponer que los argumentos de las 2 operaciones son siempre positivos.
B. Modificar la respuesta del apartado anterior, de tal forma que el reintegro de fondos a los clientes sea servido según un orden FIFO. Por ejemplo, suponer que el saldo es 200 unidades y un cliente está esperando un reintegro de 300 unidades. Si llega otro cliente debe esperarse, incluso si quiere retirar 200 unidades.

3.- Los procesos $\mathbf{P_1}$, $\mathbf{P_2}$,...,$\mathbf{P_n}$ comparten un único recurso **R**, pero sólo un proceso puede utilizarlo cada vez. Un proceso $\mathbf{P_i}$ puede comenzar a utilizar **R** si está libre; en caso contrario, el proceso debe esperar a que el recurso sea liberado por otro proceso. Si hay varios procesos esperando a que quede libre **R**, se concederá al proceso que tenga mayor prioridad. La regla de prioridad de los procesos es la siguiente: el proceso $\mathbf{P_i}$ tiene prioridad **i (1<=i<=n)**, donde los números menores implican mayor prioridad. Implementar un monitor que implemente los procedimientos para "pedir" y "liberar" el recurso.

4.- En un sistema hay dos tipos de procesos: **A** y **B**. Queremos implementar un esquema de sincronización en el que los procesos se sincronizan por bloques de 1 proceso del tipo **A** y 10 procesos del tipo **B**. De acuerdo con este esquema:

- Si un proceso de tipo **A** llama a la operación de sincronización, y no hay (al menos) 10 procesos de tipo **B** bloqueados en la operación de sincronización, entonces el proceso de tipo **A** se bloquea.
- Si un proceso de tipo **B** llama a la operación de sincronización, y no hay (al menos) 1 proceso del tipo **A** y 9 procesos del tipo **B** (aparte de él mismo) bloqueados en la operación de sincronización, entonces el proceso de tipo **B** se bloquea.
- Si un proceso de tipo **A** llama a la operación de sincronización y hay (al menos) 10 procesos bloqueados en dicha operación, entonces el proceso de tipo **A** no se bloquea y además deberán desbloquearse exactamente 10 procesos de tipo **B**.
- Si un proceso de tipo **B** llama a la operación de sincronización y hay (al menos) 1 proceso de tipo **A** y 9 procesos de tipo **B** bloqueados en dicha operación, entonces el proceso de tipo B no se bloquea y además deberán desbloquearse exactamente 1 proceso del tipo A y 9 procesos del tipo **B**.

5.- Realizar un monitor para un problema en que intervienen:

- Hay procesos productores de mensajes.
- Hay procesos consumidores de mensajes. El número es conocido para cada ejecución del problema.
- Se podrán almacenar como máximo **N** mensajes**.**

Además se tendrán en cuenta las siguientes restricciones:

- no se pierden mensajes.
- todos los consumidores leen todos los mensajes en el orden de llegada.
- los consumidores no deben esperar a otros consumidores.

RELACIÓN DE PROBLEMAS 4

Paso de Mensajes

1.- Sean dos conjuntos de 4 procesos cada uno: P(i:1..4) y Q(i:1..4). Deseamos que cada proceso P(j) envíe una secuencia de valores enteros al proceso correspondiente Q(j). En lugar de realizar esta comunicación directamente, la haremos a través de un proceso multiplexor y un proceso demultiplexor. Cuando un proceso P(j) quiere enviarle un valor a su Q(j) lo envía directamente al multiplexor, el cual enviará dicho valor junto con el índice (j) del proceso emisor al proceso demultiplexor. El proceso demultiplexor, enviará el valor transmitido al proceso Q(j) correspondiente, de acuerdo con el valor de índice recibido.

A. La comunicación será asíncrona.
B. La comunicación será síncrona.
C. Modificar la solución para que los procesos P(j) reciban la confirmación que los procesos Q(j) han recibido el mensaje.

2.- En un sistema distribuido, 6 procesos clientes necesitan sincronizarse de forma específica para realizar cierta tarea, de forma que dicha tarea sólo podrá ser realizada cuando tres procesos estén preparados para realizarla. Para ello, envían peticiones a un proceso controlador del recurso y esperan respuesta para poder realizar la tarea específica.
El proceso controlador se encarga de asegurar la sincronización adecuada. Para ello, recibe y cuenta las peticiones que le llegan de los procesos, las dos primeras no son respondidas y producen la suspensión del proceso que envía la petición (debido a que se bloquea esperando respuesta) pero la tercera petición produce el desbloqueo de los tres procesos pendientes de respuesta. A continuación, una vez desbloqueados los tres procesos que han pedido (al recibir respuesta), inicializa la cuenta y procede cíclicamente de la misma forma sobre otras peticiones.

A. La comunicación será asíncrona.
B. La comunicación será síncrona.

3.- En un sistema distribuido, 3 procesos productores producen continuamente valores enteros y los envían a un proceso buffer que los almacena temporalmente en un array local de 4 celdas enteras para ir enviándoselos a un proceso consumidor. A su vez, el proceso buffer realiza lo siguiente, sirviendo de forma equitativa al resto de procesos:

A. Envía enteros al proceso consumidor siempre que su array local tenga al menos dos elementos disponibles.
B. Acepta envíos de los productores mientras el array no esté lleno, pero no acepta que cualquier productor pueda escribir dos veces consecutivas en el búfer.

4.- Suponer un proceso productor y 3 procesos consumidores que comparten un buffer acotado de tamaño B. Cada elemento depositado por el proceso productor debe ser retirado por todos los 3 procesos consumidores para ser eliminado del buffer. Cada consumidor retirará los datos del buffer en el mismo orden en el que son depositados, aunque los diferentes consumidores pueden ir retirando los elementos a ritmo diferente unos de otros. Por ejemplo, mientras un consumidor ha retirado los elementos 1, 2 y 3, otro consumidor puede haber retirado solamente el elemento 1. De esta forma, el consumidor más rápido podría retirar hasta B elementos más que el consumidor más lento.

5.- Diseñar un proceso controlador que provoque que los dos primeros procesos que lo invoquen sean suspendidos y el tercero los despierte y así cíclicamente. Incluir en la solución también el proceso que invocan al controlador.

6.- Una serie de procesos envían un mensaje a un proceso controlador a través de un buzón. Una parte del contenido del mensaje es un valor entero “n”. Todos los procesos que envían un mensaje a dicho buzón son bloqueados hasta que otro proceso envía un mensaje a otro buzón. Los procesos bloqueados hasta el momento deben ser liberados en el orden establecido por el parámetro “n”. Cuando todos esos procesos han sido liberados, el proceso podrá continuar.

7.- Supongamos que un centro de computadores tiene dos impresoras, A y B, que son parecidas pero no idénticas. Tres clases de procesos usan las impresoras: los que usan la impresora A, los que usan la impresora B y los que usan ambas impresoras.

- Desarrollar el código que cada clase de proceso ejecuta para acceder y liberar una impresora. Construir un controlador que asigne las impresoras y que permita usarlas con el máximo rendimiento.
- Supongamos que en el sistema hay procesos de dos prioridades. Desarrollar el código que cada clase de proceso ejecuta para acceder y liberar una impresora. Construir un controlador que asigne la impresoras.

8.- Coches que vienen del norte y del sur llegan a un puente de una sola vía. Los coches en el mismo sentido pueden cruzar el puente a la vez, pero los coches en sentido opuesto no:

- Desarrollar una solución a este problema modelando los coches como procesos.
- Modificar la solución anterior para asegurar equitatividad. Por ejemplo, permitir que como mucho “C” coches pasen en una dirección si hay coches esperando en la otra dirección.

9.- Un ascensor en el que caben cuatro personas atiende las llamadas que se le hacen desde varios pisos. En estos pisos llegan personas que quieren subir o bajar a otros pisos. Suponed que se atienden sólo a las personas que van en la misma dirección que el ascensor. Desarrollar el código del proceso ascensor y el código del proceso persona.