3.- Considere un semáforo cuyo valor actual es 3. Una operación de `señal` sobre el mismo...

a) Deja un valor de 2 en el semáforo
b) Deja un valor de 4 en el semáforo
c) Deja un valor de 4 en el semáforo excepto si existen procesos en espera por el mismo, en cuyo caso queda con 3
d) Deja un valor de 4 en el semáforo y despierta a uno de los procesos en espera, si lo hay, en cuyo caso queda con 3

4.- Los semáforos son herramientas de:

a) Comunicación entre procesos *b) Sincronización entre procesos*
c) Planificación de procesos d) Todas las anteriores son ciertas

5.- En el interbloqueo, la estrategia que requiere declarar por adelantado los recursos máximos necesarios es:

a) Prevención del interbloqueo mediante negación de retención y espera
b) Prevención del interbloqueo mediante espera circular
c) Detección y recuperación
d) Evitación mediante el algoritmo del banquero

1.- **(3 puntos)** Se pide escribir un programa que conste de dos procesos concurrentes A y B sincronizados mediante dos semáforos de tal manera que el resultado final de la ejecución sea que los procesos escriban en la salida estándar en secuencia lo siguiente:

- Primero, el proceso A debe escribir: "Hola soy A".
- Segundo, el proceso B debe escribir: "Hola soy B".
- Tercero, el proceso A debe escribir: "Se despide A".
- Cuarto, el proceso B debe ecribir: "Se despide B".

Solución:

```
Program/Module Presentación;
var
   A_continua, B_continua: semaforo;

Process A;
begin
   put("Hola soy A");
   signal(B_continua);
   wait(A_continua);
   put("Se despide A");
   signal(B_continua);
end;

Process B;
begin
   wait(B_continua);
   put("Hola soy B");
   signal(A_continua);
   wait(B_continua);
   put("Se despide B");
end;

begin
   inicializa(A_continua, 0);
   inicializa(B_continua, 0);
   cobegin
      A, B;
   coend;
end;
```

4.- El siguiente código pretende gestionar la exclusión mutua en el acceso a una sección crítica, usando la instrucción atómica `swap(a, b)` que intercambia los valores de `a` y `b`:

```
flag:=?;
repeat
swap(bloqueo, flag);
until flag=false;
/*sección crítica*/
bloqueo:=?
```

¿Con qué valores deben sustituirse las interrogaciones para que funciones adecuadamente?

a) flag=false y bloqueo=false b) flag=false y bloqueo=true c) *flag=true y bloqueo=false* d) flag=true y bloqueo=true

5.- El algoritmo del banquero...

a) requiere la declaración previa de cuántos procesos se van a ejecutar.
b) es un algoritmo de detección de interbloqueo
c) puede retener la ejecución de un proceso, aun cuando existan recursos disponibles para él.
d) sólo es implementable si el número de recursos es ilimitado

2.- Un proceso en estado de bloqueo

a) siempre espera por la ocurrencia de un evento
b) siempre espera por la conclusión de una actividad de E/S
c) reside en la cola de preparados en espera de entrar nuevamente en la CPU
d) reside en la cola de procesos de baja prioridad

3.- Indique si las siguientes afirmaciones son verdaderas:

I. La técnica de envejecimiento en la estrategia de planificación por prioridades se utiliza para evitar el interbloqueo.
II. En los algoritmos de planificación de tipo expropiativo el proceso que se está ejecutando puede ser interrumpido por parte del sistema operativo

a) I: sí, II: sí. b) I: sí, II: no. *c) I: no, II: sí.* d) I: no, II: no

4.- Considere un semáforo cuyo valor actual es 6. Una operación de `señal` sobre el mismo...

a) Deja un valor de 5 en el semáforo
b) Deja un valor de 7 en el semáforo
c) Deja un valor de 7 en el semáforo excepto si existen procesos en espera por el mismo, en cuyo caso queda con 6
d) Los valores de un semáforo sólo pueden ser de 0 o 1

1.- La ordenación lineal de recursos actúa sobre la condición de:

a) No apropiación.
b) Retención y espera.
c) Espera circular.
d) Exclusión mutua.

2.- El problema del interbloqueo queda resuelto con:

a) Los semáforos.
b) Las secciones críticas.
c) Variables compartidas.
d) Ninguna opción anterior.

3.- Una solución correcta al problema de la sección crítica:

a) debe garantizar la exclusión mutua.
b) debe garantizar que un proceso no interesado en entrar en sección crítica no impida a otros procesos interesados entrar en ella.
c) a) y b) son ciertas.
d) a) es cierta y b) lo sería sólo cuando se desea obtener una mayor velocidad en la gestión de la entrada en sección crítica.

4.- La operación ESPERA ó WAIT sobre un semáforo

a) no ha de ser atómica si se trata de un semáforo binario.
b) duerme siempre al proceso llamador hasta que otro proceso ejecute una SEÑAL o SIGNAL sobre el mismo semáforo.
c) a) y b) son ciertas.
d) b) es falsa.

5.- Cuando un proceso hace una operación SEÑAL ó SIGNAL sobre un semáforo general

a) Siempre se incrementa en una unidad el valor del semáforo.
b) Se incrementa en una unidad el valor del semáforo y se despierta a un proceso.
c) Se incrementa en una unidad el valor del semáforo cuando no hay procesos suspendidos.
d) Se incrementa en una unidad el valor del semáforo cuando hay procesos suspendidos.

1.- (3 puntos) En el parque de atracciones se ha establecido límite de usuarios por razones de seguridad (para ello se han instalado unos torniquetes a la entrada y a la salida que controlan en todo momento el número de personas en el parque) de forma que cuando está completo se debe esperar para entrar que alguien salga. En los días de vacaciones se forman largas colas, por lo que el parque ha decidido crear una entrada VIP, más cara que la NORMAL, pero queda preferencia a la hora de acceder a las instalaciones cuando el parque está lleno. Realizar el código de entrada y salida del parque, de forma que cuando alguien abandona el parque, su lugar será ocupado por una persona con entrada VIP, si no hubiera personas con entrada VIP esperando, entonces su lugar será ocupado por una persona con entrada NORMAL. Definir los procesos Entradas y Salidas que se ejecutan cuando una persona entra o sale del parque. Utilizar semáforos para gestionar la capacidad del parque y la cola de espera.

Solución:

```
Program/module Parque_Atracciones;

var sem_vip, sem_normal, mutex: semaforo;
    usuarios, usuarios_normales, usuarios_vip: integer;
    tiket: {vip, normal};
Const capacidad N
```

```
Process EntradaX (tiket: tiket)
  begin
     espera(mutex);
     if usuarios >= capacidad then
        if tiket = vip then
           begin
             usuarios_vip= usuarios_vip +1;
             señal(mutex);
             espera(sem_vip);
           end
        else
           begin
             usuarios_normales= usuarios_normales +1;
             señal(mutex);
             espera(sem_normal);
           end;
    else
      begin
        usuarios= usuarios+1;
        señal(mutex);
      end;
 end;
```

```
Process SalidaX(tiket: tiket)
begin
       espera(mutex);
       if usuarios_vip >= 0 then
          begin
            señal(sem_vip);
            usuarios_vip= usuarios_vip - 1;
          end;
       else
          if usuarios_normales >= 0 then
             begin
               señal(sem_normal);
               usuarios_normales= usuarios_normales - 1;
             end;
          else
             usuarios = usuarios - 1;
      señal(mutex);
   end;
```

```
Process Padre
   begin
      inicializa(mutex, 1);
      inicializa(sem_vip, 0);
      inicializa(sem_normal, 0);
      usuarios=usuarios_normales=usuarios_vip=0;
      cobegin
        EntradaS, SalidaS;
      coend
   end;
```

3.-Indique si las siguientes afirmaciones sobre semáforos son verdaderas:

I. Causan pérdidas de tiempo debido a esperas ocupadas o esperas activas.

II. Permiten realizar la sincronización de procesos.

a) I: sí, II: sí. b) I: sí, II: no. **c) I: no, II: sí.** d) I: no, II: no.

4.- Indique si las siguientes afirmaciones son verdaderas. Si un proceso está dentro de una sección crítica controlada por semáforo:

I. Ningún otro proceso puede entrar a esa sección crítica.

II. No puede realizar otra operación de *wait* o *espera* sobre otro semáforo.

a) I: sí, II: sí. **b) I: sí, II: no.** c) I: no, II: sí. d) I: no, II: no.

5.- Indique si las siguientes afirmaciones son verdaderas:

I. Un estado inseguro siempre conduce a interbloqueo.

II. Las técnicas de evitación de interbloqueos se basan en evitar que se cumplan algunas de las condiciones necesarias para que se produzca interbloqueo.

1.- Un proceso en estado bloqueado

a) siempre espera por la ocurrencia de un evento.

b) siempre espera por la conclusión de una actividad de E/S.

c) reside en la cola de preparados en espera de entrar nuevamente en la CPU.

d) reside en la cola de procesos de baja prioridad.

2.- La utilización de una cola de procesos para la implementación de los semáforos:

a) es imprescindible para colocar en ella a los procesos que ejecutan la operación *wait* o *espera*.

b) se resuelve gracias a las instrucciones atómicas (test-and-set ó swap).

c) evita las soluciones con espera activa.

d) todas las anteriores son ciertas.

3.- El problema de la sección crítica aparece porque

a) existen sistemas de memoria compartida aparte de los de paso de mensajes.

b) la espera activa es insuficiente para resolver la exclusión mutua de procesos concurrentes.

c) la ejecución de porciones de código del sistema operativo accediendo a tablas protegidas ha de efectuarse en exclusión mutua.

d) varios procesos concurrentes pueden acceder a un mismo conjunto de datos.

4.- ¿Cuál es el número mínimo de procesos y recursos necesarios para que se forme interbloqueo?

a) Un proceso y una instancia de recurso.

b) Un proceso y dos instancias de recurso.

c) Dos procesos y una instancia de recurso.

d) Dos procesos y dos instancias de recurso.

5.- Dos procesos que tienen secciones críticas diferentes:

a) Pueden acceder a la vez a las secciones críticas.

b) No pueden acceder a la vez a las secciones críticas.

c) Pueden acceder a la vez a las secciones críticas si se establece comunicación entre los dos procesos.

d) Pueden acceder a la vez a las secciones críticas si se utilizan semáforos no binarios.

1.- (2 puntos) Una tribu de caníbales cenan en comunidad una gran olla que contiene M exploradores cocinados. Cuando un caníbal quiere comer, él mismo se sirve de la olla un explorador, a menos que esté vacía. Si la olla está vacía, el caníbal despierta al cocinero y espera a que éste llene la olla. Desarrollar el código de las acciones de los caníbales y el cocinero usando semáforos.
Solución:

```
program cena;
var
olla, i:integer;
mutex, comer,cocina: semáforo;

process canibalX;
begin
  while true
  begin
    wait(mutex)
    if olla=0 then
    begin
      signal(cocina);
      wait(comer);
    end;
    olla:=olla-1;
    signal(mutex);
  end;
end,

process cocinero;
begin
  while true
  begin
    wait(cocina);
    olla:=m;
    signal(comer);
  end;
end;


begin
  olla:=m,
  inicializa(mutex,1);
  inicializa(cocina,0);
  inicializa(comer,0);
  cobegin
    canibales;
    cocinero;
  coend;
end;
```

1.- Dos procesos que tienen secciones críticas diferentes:
a) Pueden acceder a la vez a las distintas secciones críticas.
b) No pueden acceder a la vez a las distintas secciones críticas.
c) Pueden acceder a la vez a las secciones críticas si se utilizan semáforos no binarios.
d) Pueden acceder a la vez a las secciones críticas si se establece comunicación entre los dos procesos.

2.- El número máximo de procesos que pueden estar bloqueados en un semáforo binario son:
a) Uno.
b) Dos.
c) Depende de la capacidad que tenga la cola del semáforo.
d) Depende de la declaración del semáforo.

1.- (3 puntos) En un parque infantil hay un tobogán y un tren móvil. Los niños comparten las atracciones pero éstas tienen una capacidad limitada. Todos los niños quieren inicialmente montar en el tobogán y, después, montar en el tren. Pero se encuentran con el inconveniente de que sólo uno de ellos puede montar en el tobogán al mismo tiempo y sólo tres pueden montar en el tren. Define un proceso que ejecuten los niños concurrentemente de forma que se sincronicen estas actividades usando semáforos.

```
module Niños_felices
var
  semaphore: tren {general}
             tobogan{binario}

Process NiñosX;
  begin
    loop
      begin
        wait(tobogan);
        montar_tobogan();
        signal(tobogan);
        wait(tren);
        montar_en_tren();
        signal(tren);
      end;
  end;

Process Padre;
  begin
inicializa (tren=3);
inicializa (tobogan=1);

  cobegin
    Niños;
  coend;
end;
```

Para las tres preguntas siguientes, considérese los procesos Productor y Consumidor, que se describen a continuación, y que se ejecutan concurrentemente.
El semáforo buffer_vacio indica el número de registros que están vacíos. El semáforo buffer_lleno indica el número de registros que están llenos. N es el número de registros del buffer

```
semaphore: buffer_lleno, buffer_vacio, exclusión;
buffer_lleno=N-1; buffer_vacio=1;

Process Productor;
  begin
    wait(buffer_vacio);
    wait(exclusion);
    Produce;
    signal(exclusion);
    signal(buffer_lleno);
  end;

Process Consumidor;
  begin
    wait(buffer_lleno);
    wait(exclusion);
    Consume;
    signal(exclusion);
    signal(buffer_vacio);
  end;
```

1.- En la codificación realizada, el semáforo exclusión:
a) Se debe inicializar a 0. **b) Se debe inicializar a 1.** c) Se debe inicializar a N. d) No se debe inicializar.

2.- En la codificación realizada, ¿Qué proceso entra primero en la sección crítica?.
a) El productor. b) Los dos se quedan bloqueados. **c) No se puede determinar.** d) El consumidor.

3.- Si se ejecutan tres procesos productores seguidos de un consumidor, la situación de las variables es:
a) exclusión a 1, buffer_lleno=N, buffer_vacio=0. b) exclusión a 0, buffer_lleno=N, buffer_vacio=0.
c) exclusión a 1, buffer_lleno=N, buffer_vacio=0 y un productor en la cola del semáforo buffer_vacio.
d) exclusión a 0, buffer_lleno=N, buffer_vacio=0 y un productor en la cola del semáforo buffer_vacio.

1.- Tanto en los semáforos binarios como en los generales:
a) La cola debe ser FIFO.
b) En los semáforos binarios la cola debe ser FIFO y en los generales de cualquier tipo.
c) En ambos tipos de semáforos la cola puede ser gestionada por un algoritmo FIFO u otro distinto.
d) En ninguno de los dos tipos de semáforos se utilizan colas.

2.- Las operaciones wait y signal:
a) Son atómicas en todos los sistemas operativos.
b) En unos sistemas operativos son atómicas y en otros no.
c) El programador debe encargarse de que sean atómicas.
d) no son nunca atómicas.

3.- La ventaja en el uso de los monitores frente a los semáforos es que:
a) No se produce espera activa.
b) No se produce interbloqueo.
c) La sincronización está implícita, basta con invocar el procedimiento del monitor.
d) La exclusión mutua está implícita, basta con invocar el procedimiento del monitor.

1.- (3 puntos) Los directores de un complejo turístico nos piden que regulemos con semáforos el acceso de los turistas a los templos a visitar desde una sala de exposiciones. Para acceder a los templos es necesario esperar en la sala a que venga a buscarnls un guía. El número total de guías es G. Si un visitante quiere acceder a los templos y hay más gente esperando a que venga un guía, deberá hacer cola. La visita es personalizada, es decir, cada guía se lleva sólo a un visitante. Si un guía esta disponible y no hay visitantes esperando a la visita de los templos, el guía descansará.

```
module Visita_Templos
var
  semaphore: guia {general}
             visitante{binario}

Process VisitanteX;
  begin
    loop
      begin
        signal(visitante);
        wait(guia);
        seguir al guía;
        ver los templos;
      end;
end;

Process guiaX;
    begin
      loop
      begin
        wait(visitante);
        signal(guia);
        explicar el templo;
      end;
  end;

Process Padre;
begin
inicializa (visitante,0)
inicializa (guia, G);
cobegin
visitantes;
guias;
coend;
end;
```

6.- En los monitores se cumple que

a) La exclusión mutua no está implícita.
b) No proporcionan por si mismos un mecanismo para la sincronización de tareas.
c) Las dos afirmaciones anteriores son verdaderas.
d) Ninguna de las anteriores.

Solución: pp.124-125 del libro base de la asignatura

Un *monitor* es un conjunto de procedimientos que proporcionan el acceso con exclusión mútua a un recurso o conjunto de recursos (datos o dispositivos) compartidos por un grupo de procesos. La ventaja para la exclusión mútua que presenta un monitor frente a los semáforos u otro mecanismo es que ésta está ahora implicita. Por otra parte, los monitores no proporcionan por si mismos un mecanismo para la sincronización de tareas y, por ello, su construcción debe completarse permitiendo, por ejemplo, que se puedan usar señales para sincronizar los procesos.

Respuesta:B) **No proporcionan por si mismos un mecanismo para la sincronización de tareas**

7.- Decir si las siguientes afirmaciones relativas a un semáforo S inicializado con un valor N son ciertas:

I) Si N=1 la ejecución de una operación espera(S) por parte de un proceso provocará la suspensión de dicho proceso y su colocación en la cola de tareas en espera.
II) Si N=3 la ejecución de una operación señal(S) deja un valor de 4 en el semáforo.

a) I: si, II: si. b) I: si. II: no. c) I: no, II: si. d) I: no, II: no.

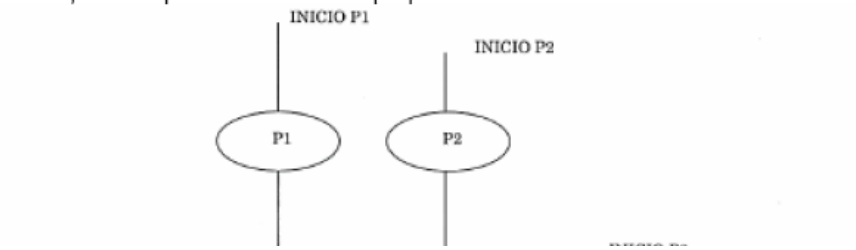
Solución: pp.122 del libro base de la asignatura

Afirmación I) Si un semáforo S está inicializado con un valor N=1 entonces la ejecución de una operación espera(S) por parte de un proceso simplemente producirá que el semáforo tome el valor N=0 pero no provoca la suspensión del proceso. Luego esta afirmación es FALSA.

Afirmación II) Si un semáforo S está inicializado con un valor 3 entonces la ejecución de una operación señal (S) por parte de un proceso producirá que el semáforo tome el valor 4. En conclusión la afirmación es VERDADERA.

Respuesta:**C) I: no, II: si**.

1.- (3 puntos) Las aplicaciones de gran tamaño suelen estar compuestas de varios procesos. Hay veces que dos o más procesos se pueden ejecutar en paralelo hasta un determinado instante. Luego, para que puedan continuar cada uno de los procesos que se han estado ejecutando en paralelo, es necesario que se encuentren en un punto. Vamos a suponer que nuestro sistema es de tres procesos y los puntos de encuentro están dados en la figura adjunta. Se pide modelar con semáforos esta situación de forma que hasta que dos procesos no hayan llegado al punto donde se debe encontrar, ninguno de ellos puede continuar. El proceso que antes llegue al punto de encuentro deberá esperar a que el otro alcance dicho punto. No se sabe cuál de los dos procesos comienza a ejecutarse primero o cual es el que primero termina.



6.- ¿Existe alguna diferencia entre la operación de espera de un semáforo y la de una variable de condición de un monitor?

a) No.
b) Si.
c) Depende de la implementación del monitor y del semáforo.
d) No es posible saberlo sin tener más datos.

Solución: pp. 163 del libro base de la asignatura.

La ejecución de la operación de espera de una variable de condición siempre suspende al proceso que la emite mientras que la de un semáforo depende del valor del indicador. Luego si existe diferencia.

Respuesta: **B) Si.**

```
Process P1;
   begin
     instrucciones de P1;
     signal(semP2P1);
     wait(semP1P2);
     instrucciones de P1;
     signal(semP3P1);
     wait(semP1P3);
     instrucciones de P1;
    end;
```

```
Process P2;
   begin
     instrucciones de P2;
     signal(semP1P2);
     wait(semP2P1);
     instrucciones de P2;
     signal(semP3P2);
     wait(semP2P3);
     instrucciones de P2;
    end;
```

```
Process P3;
   begin
     instrucciones de P3;
     signal(semP2P3);
     wait(semP3P2);
     instrucciones de P3;
     signal(semP1P3);
     wait(semP3P1);
     instrucciones de P3;
    end;
```

```
process Padre;
begin
semP1P2=0, semP1P3=0, semP2P1=0, semP2P3=0, semP3P1=0, semP3P2=0;
cobegin
P1, P2, P3;
coend;
end;
```