Práctico 2. Sincronización

1. Defina
   a) Tareas
   b) Prioridades fijas y dinámicas
2. Crear dos threads con prioridades fijas iguales que impriman sucesivamente por la salida estándar, las palabras "Tiempo" y "Real" respectivamente.
3. Crear dos threads con prioridades fijas que modifiquen una variable entera global (inicialmente igual a 0), sumando 1 y restando 1 respectivamente.
   a) Prioridades iguales.
   b) Prioridades distintas.
   c) Analizar.
4. Defina
   a) Región Crítica
   b) Mutex
   c) Semáforos
   d) Variables de Condición
5. Sincronizar el acceso a un buffer compartido de 12 lugares por parte de dos threads que se activan periódicamente cada 1000 milisegundos, donde uno establece un valor 0 (cada 100 milisengundos) en todo el buffer y el otro un valor 1 (cada 100 milisengudos) en todo el buffer. El objetivo es mantener invariante todo el hilo.
   a) Implementar
   b) Cuantos semáforos son necesarios?
   c) Quitar los semáforos y analizar el problema.
6. Sincronización múltiple: Se tienen los threads A, B y C. Todos los threads se ejecutan con un período de 150 milisegundos y al llegar a un punto en concreto de su ejecución no pueden seguir hasta que los otros dos procesos no hayan llegados a sus puntos de sincronización correspondiente.

   Suponer que desde su inicio hasta llegar al punto de sincronización: A tarda 10 milisegundos, B tarda 100 milisegundos y C tarda 20 milisegundos.
7. Implementar el problema del productor-consumidor sobre un buffer circular FIFO. El productor, pone elementos en un buffer siempre que haya lugar disponible, en caso contrario espera. El consumidor, consume elementos del mismo buffer siempre que haya elementos disponibles, en caso contrario espera.
8. Existen cinco filósofos que emplean su tiempo entre la meditación y la comida. Se encuentran sentados en una mesa circular en cuyo centro hay un plato de arroz y disponen de 5 palillos situados en medio de cada dos de ellos. El principal problema radica en que para comer son necesarios dos palillos, con lo que no podrán comer todos los filósofos a la vez. Para resolver el problema correctamente, cada filósofo solo puede comer si se encuentran libres sus dos palillos adayacentes (aunque estuvieran libres los otros 3 palillos), y suponiendo, como es lógico, que el tiempo que dedica un filósofo a comer es limitado, también se puede pedir que ningún filósofo se muera de hambre.

a) Implementar suponiendo que,

```
N=5 (constante)
palillos[N] (tipo semáforo)
filosofo(f){
while(1){
   MEDITA
   wait(palillos[i]) //espera por el palillo izquierdo hasta conseguirlo
   wait(palillos[i+1]) //espera por el palillo derecho hasta conseguirlo
   COME
   signal(palillos[i]) //libera el palillo izquierdo
   signal(palillos[i+1]) //libera el palillo derecho
   }
   }
```

Analizar, explicar fallos y dar posibles soluciones.

b) Implementar suponiendo que:

```
N=5 (constante)
tipo_estado={comiendo, esperando_comer, meditando}
estado[N] (tipo tipo_estado)
espera[N] (tipo semáforo)
mutex (tipo semáforo)
filosofo(j){
MEDITA
wait(mutex)
if estado[j+1]=comiendo or estado[j-1]=comiendo
   then {
   estado[j]=esperando_comer;
   signal(mutex)
   wait(espera[j])
   }
   else estado[j]=comiendo
signal(mutex)
COME
if estado[j+1]=esperando_comer and estado[j+2]<>comiendo
   then {
   estado[j+1]=comiendo
   signal(espera[j+1])
   }
if estado[j-1]=esperando_comer and estado[j-2]<>comiendo
   then {
   estado[j-1]=comiendo
   signal(espera[j-2])
   }
   signal(mutex);
}
```

Analizar, explicar fallos y dar posibles soluciones.

c) Implementar suponiendo que

```
N=5 (constante)
tipo_estado={comiendo, esperando_comer, meditando}
estado[N] (tipo tipo_estado)
espera[N] (tipo semáforo)
mutex (tipo semáforo)
filosofo(j){
MEDITA
```

```
estado[j]=esperando_comer
wait(mutex)
if estado[j+1]<>esperando_comer or estado[j+1]<>meditando or estado[j-
1]<>esperando_comer or estado[j+1]<>meditando
   then {
   signal(mutex)
   wait(espera[j])
   }
   else{
   estado[j]=comiendo
   signal(mutex)
   }
COME
wait(mutex)
estado[j]=meditando
if estado[j+1]<>esperando_comer and estado[j+2]<>comiendo
   then {
   estado[j+1]=comiendo
   signal(espera[j+1])
   }
if estado[j-1]<>esperando_comer and estado[j-2]<>comiendo
   then {
   estado[j-1]=comiendo
   signal(espera[j-2])
   signal(mutex);
   }
}
```

Analizar, explicar fallos y dar posibles soluciones.

d) Implementar suponiendo que

```
N=5 (constante)
tipo_estado={comiendo, esperando_comer, meditando}
estado[N] (tipo tipo_estado)
espera[N] (tipo semáforo)
mutex (tipo semáforo)
filosofo(j){
MEDITA
wait(mutex)
if estado[j+1]=comiendo and estado[j+1]=comiendo
   then {
   wait(espera[j])
   signal(mutex)
   }
   else{
   estado[j]=comiendo
   signal(mutex)
   }
COME
estado[j]=meditando
if (estado[j+2]<>meditando or estado[j+2]<>esperando_comer) and estado[j+
1]=esperando_comer
   then {
   estado[j+1]=comiendo
   signal(espera[j+1])
   signal(mutex)
```

```
    }
if (estado[j − 2]<>meditando or estado[j − 2]<>esperando_comer) and estado[j −
1]=esperando_comer
    then {
    estado[j − 1]=comiendo
    signal(espera[j − 1])
    signal(mutex)
    }
}
```

Analizar, explicar fallos y dar posibles soluciones.

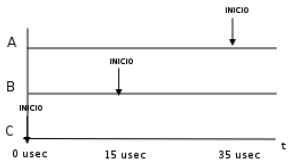
9. Sea la sucesión en que cada término es la suma de los dos anterios: $a_1 = 0$; $a_2 = 1$; $a_3 = 1$;... $a_i = a_{i-2} + a_{i-1}$. Para calcular las sumas parciales de los términos impares ($a_1 + a_3 + a_5 + a_7 \ldots$) de esta sucesión, tenemos dos tareas, SUCESION y SUMA. La primera calcula los términos de la sucesión y la segunda deberá calcular la suma de los términos impares. Ambas utilizan la variable compartida PASAR.

10. Implementar el problema de los lectores-escritores. Una base de datos es compartida por varios procesos, distinguiéndose entre ellos dos tipos: los lectores que sólo nquieren leer la base de datos, y los escritores que quieren actualizar dicha base. Obviamente, si dos o más lectores acceden a la base de forma simultánea no existirá ningún conflicto, sin embargo, si un escritor y algún otro escritor o lector acceden a la base de datos de manera simultánea, entonces si puede producir conflictos. Existen dos variantes: que se le de prioridad a los lectores frente a los escritores (los lectores solo esperaran cuando exista un escritor escribiendo), o que se le de prioridad a los escritores frente a los lectores (si un escritor está esperando ningún lector puede comenzar a leer).

11. Se tienen las siguientes tareas:

    - A de prioridad 1 y con una duración de 30 useg. A los 10 useg ingresa en su región crítica, en la cual utiliza un recurso X.
    - B de prioridad 2 y con una duración de 40 useg. A los 10 useg ingresa en su región crítica, en la cual utiliza un recurso Y.
    - C de prioridad 3 y con una duración de 30 useg. A los 10 useg ingresa en su región crítica, en la cual utiliza un recurso X.

    La tarea C es la primera en iniciar a los 0 useg. A los 15 useg, da inicio la tarea B y 20 useg después, inicia la tarea A.



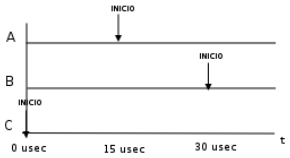
	Duracion	Intervalo	Sec Crítica
A	30 useg	(35,90] useg	[X,20 useg]
B	40 useg	(15,90] useg	[Y,30 useg]
C	30 useg	(0,90] useg	[X,20 useg]

    Inplementar y explicar el resultado.

12. Se tienen las siguientes tareas:

    - A de prioridad 1 y con una duración de 30 useg. A los 10 useg ingresa en su región crítica, en la cual utiliza un recurso X.
    - B de prioridad 2 y con una duración de 40 useg. A los 10 useg ingresa en su región crítica, en la cual utiliza un recurso Y.
    - C de prioridad 3 y con una duración de 30 useg. A los 10 useg ingresa en su región crítica, en la cual utiliza un recurso X.

La tarea C es la primera en iniciar en 0 useg. A los 15 useg, da inicio la tarea A y 15 useg después, inicia la tarea A.



	Duracion	Intervalo	Sec Crítica
A	30 useg	(15,90] useg	[X,20 useg]
B	40 useg	(30,90] useg	[Y,30 useg]
C	30 useg	(0,90] useg	[X,20 useg]

Inplementar y explicar el resultado. Comparar con el ejercicio 11.