Departamento de Programación
Facultad de Informática
Universidad Nacional del Comahue

Programación Concurrente
Trabajo Práctico OBligatorio 1
Preparación 3 para examen final libre

Deben resolver cada ejercicio aplicando un único mecanismo de sincronización (1 con semáforos, 1 con monitores y 1 con locks)

Problema 1: Actividades Riesgosas

Hay un salón donde se practican distintas actividades de acrobacia aérea. Acro telas, Lyra acrobática y yoga en aro.
Cada actividad tiene cupo para 4 personas. Y cada persona puede optar por realizar dos de ellas por turno, puesto que cada actividad es de media hs. Cada turno es de 1hs y tiene capacidad para 12 personas.
La actividad comienza cuando están los cupos llenos de las 3 actividades. Luego, cuando rotan de actividad, puede ocurrir que más de 4 personas quieran realizar una actividad, en ese caso, la persona debe optar por una de las actividades que queden disponibles, evitando (en lo posible) realizar la misma actividad inicial.

Por ejemplo:
1er TURNO

personas por turno	30 min	30 min
p1	a1	a3
p2	a3	a1
p3	a2	a1
p4	a3	a1
p5	a1	a2
p6	a3	a1
p7	a2	a3
p8	a3	a1
p9	a1	a3
p10	a2	a1
p11	a2	a2
p12	a1	a2

En este caso hay 6 personas que quieren realizar como 2da actividad, la actividad a1, por lo tanto 2 de ellas deben realizar las actividades que quedan con lugares vacíos, puesto que solo hay lugar para 4 personas por actividad. Es decir, p_i va a a2 y p_j a a3.

Problema 2: Trabajadores Incansables

Un restaurante realiza pizzas para enviar a domicilio, y existen *maestros pizzeros* y *repartidores*. Cada maestro pizzero se dedica a preparar un tipo de pizza (o sea prepara napolitanas o prepara veganas) y las prepara según van llegando pedidos. Los pedidos pueden ser de 1 pizza napolitana o 2 pizzas veganas.

Cada vez que un pedido está completo (es decir se prepararon las pizzas solicitadas), el cocinero lo deja en el mostrador a disposición de un repartidor para su reparto. A su vez, cada repartidor espera a que se complete algún pedido (de 1 o 2 pizzas), lo retira y se la lleva al cliente correspondiente. Tras ello regresa a la pizzería y espera por un nuevo pedido.

Simule la entrega al cliente considerando el traslado y un mensaje apropiado.

El mostrador tiene una capacidad limitada: no puede haber más de *MAX* pedidos pendientes de reparto. Cada 10 viajes que haga un repartidor se sienta a descansar comiendo una pizza hecha por algún cocinero.

Considere (e implemente) que existe un hilo "generador de pedidos" que genera aleatoriamente la información de cada pedido: tipo de pizza y nombre del cliente
Simule la situación presentada utilizando un mecanismo de sincronización adecuado.

Problema 3 Buffer oscilante.

En algunos sistemas de buffering las llamadas a las operaciones de insertar y extraer se realizan por rachas. En estas situaciones es deseable la posibilidad de insertar y extraer datos sobre el mismo buffer de forma simultánea.
En general, no se puede asumir que la ejecución simultánea (sin asegurar exclusión mutua) de dos operaciones cualesquiera sobre un recurso del tipo cola deje al recurso en un estado consistente, por lo tanto los accesos sobre dicho recurso deben ser excluyentes.

> ¿Qué ocurriría si el buffer estuviera formado por dos colas independientes? Pareciera que existe la posibilidad de insertar en una y extraer de la otra de forma simultánea, sólo es necesario establecer un protocolo que asegure que el buffer, en su conjunto, respeta una política FIFO. Dicho protocolo consiste en decidir (oscilar) entre una cola u otra en cada momento. Veamos un escenario de ejecución:

a. Supongamos las dos colas inicialmente vacías, una de ellas etiquetada para insertar y otra para extraer.

(insertar) 1 | (extraer) 2

b. Si un proceso quiere extraer datos no puede hacerlo porque no hay ningún dato (y más precisamente porque no hay datos en la cola de extraer).

c. Si un proceso quiere insertar datos tendría que hacerlo en la cola etiquetada para ello (la cola 1).
Supongamos la inserción de un dato d1. Además de insertar d1 en la cola 1 el proceso debería "oscilar" las colas, es decir, cambiar las etiquetas de insertar y extraer para posibilitar la simultaneidad entre una extracción y una inserción.

(extraer) 1 d1 | (insertar) 2

d. Supongamos que llegan dos procesos que quieren insertar datos (d2 y d3), tendrán que insertar en la cola etiquetada para ello y lo tendrán que hacer en exclusión mutua:

(extraer) 1 d1 | (insertar) 2 d2 d3

e. Si ahora un proceso quiere insertar y otro extraer pueden realizar la operación de forma simultánea y, finalmente . . . , hay que volver a "oscilar":

(insertar) 1 | (extraer) 2 d2 d3 d4

Como puede observarse la situación permite que de nuevo puedan ejecutarse de forma simultánea una operación de inserción y una de extracción.

Como puede observarse en los ejemplos, ***la oscilación debe producirse cada vez que la cola de extracción queda vacía.***

Se debe implementar un sistema de buffering como el indicado, considerando *hilos insertores* e *hilos extractores,* y el recurso compartido. Considere ***tamaño ilimitado***