Programación Concurrente 2018
Trabajo Práctico Nº 5:

1. Retome el ejercicio 3 del práctico 4 y ahora considere:

   (a) que hay un buffer de tamaño *kA* compartido por las impresoras de tipo A y un buffer de tamaño *kB* compartido por las impresoras de tipo B.

   (b) ¿que podría suceder si existiera un tercer buffer compartido por todas las impresoras para los trabajos a los que les sirve cualquier tipo de impresora?

2. Considere el problema de los lectores/escritores e implementarlo:

   (a) con monitores

   (b) con semáforos generales

3. En un cuartel hay un comedor para 100 soldados. El soldado cuando quiere comer entra en el recinto y toma una bandeja con comida en uno de los 5 mostradores que existen para tal efecto; la bandeja tiene un vaso de agua o un botellín de refresco, si escoge esto ltimo necesita uno de los 10 abridores. Si quiere postre se dirige a uno de los 3 mostradores que lo despachan. Cuando finaliza la comida sale del recinto.

   Realizar un programa concurrente de forma que utilizando algún mecanismo de sincronización coordine las tareas de los soldados.

4. Coches que vienen del norte y del sur pretenden cruzar un puente sobre un río. Solo existe un carril sobre dicho puente(de una sola vía, es decir que los coches en la misma dirección no pueden adelantarse). Por lo tanto, en un momento dado, sólo puede ser cruzado por uno o más coches en la misma dirección (pero no en direcciones opuestas).

   (a). Implementar la solución con semáforos.
   (b). Implementar teniendo en cuenta el código del siguiente monitor:

```
MONITOR GestionaTrafico
.....
        PROCEDIMIENTO EntrarCocheDelNorte
        .....
        PROCEDIMIENTO SalirCocheDelNorte
        .....
        PROCEDIMIENTO EntrarCocheDelSur
        .....
```

PROCEDIMIENTO SalirCocheDelSur

Que resuelva el problema del acceso al puente suponiendo que llega un coche del norte (sur) y cruza el puente si no hay otro coche del sur (norte) cruzando el puente en ese momento.

(c). Mejorar la implementación del monitor anterior, de forma que la dirección del trafico a través del puente cambie cada vez que lo hayan cruzado 10 coches en una dirección, mientras 1 o más coches estuviesen esperando cruzar el puente en dirección opuesta.

(d). Realizar una clase test de prueba a fin de que las implementaciones antes solicitadas pueda ser probadas.

IMPORTANTE: debe considerar que la implementación asegure que los coches crucen el puente en el orden en que llegaron, es decir, si C1 llego al puente antes que C2, en la misma dirección, C1 debe poder empezar y terminar de cruzar el puente antes que C2.

5. **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.

¿Que 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

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

6.- **Pasteleria**

Una pasteleria tiene tres hornos que producen tres tipos diferentes de pasteles: A, B y C, con pesos diferentes: pesoA, pesoB y pesoC. Procedentes de los hornos, los pasteles se van situando en un *mostrador* común.

Los pasteles son empaquetadoss en cajas. Uno o varios robots *Empaquetador* toman pasteles del mostrador y los introducen en la caja (ver figura). Cada caja puede contener un número diferente de pasteles siempre y cuando no se sobrepase un peso límite, denominado PesoMaximo. Por este motivo, antes de incluir un pastel en la caja, cada empaquetador debe asegurarse de que con su inclusión no se sobrepasa el peso máximo. Si no se sobrepasa el peso se incluye el pastel en la caja; en otro caso, un *Brazo* mecánico se encarga de retirar la caja que se estaba llenando y posteriormente la sustituye por una caja vacía.

Tener en cuenta de que se trata de llenar cada caja lo más posible, lo cual puede ser conseguido por uno cualquiera de los robots que intentan depositar simultáneamente algun pastel, de pesos que pueden variar. Se considera que no hay interferencia física entre robots que intentan soltar ubicar pasteles al mismo tiempo en la caja.

Se asume que inicialmente hay una caja vacía junto al mostrador.

Considere las siguientes operaciones:

Retirar Caja : *hace que el proceso que lo invoca quede bloqueado hasta que la caja que estaba siendo llenada es retirada por el brazo auxiliar . Requiere que haya una caja en la zona de rellenado .*

Reponer Caja: *hace que el proceso que lo invoca quede bloqueado hasta que el brazo auxiliar coloque una caja vacía en el área de rellenado. No debe haber ninguna caja en la zona de rellenado .*

Tomar Pastel, retorna Peso : *provoca que el proceso que lo invoca quede bloqueado hasta que el empaquetador toma el pastel más cercana del mostrador, indicando* Peso *el peso del mismo.*

Soltar Pastel : *provoca que el proceso que lo invoca quede bloqueado hasta que el empaquetador suelta el pastel que acaba de tomar en la caja del área de rellenado. Es necesario que haya una caja en el área de rellenado, que el robot empaquetador en cuestión tenga un pastel y que la inclusión de ese pastel en la caja no haga sobrepasar el peso máximo permitido. Físicamente, no hay interferencia entre dos robots que intentan depositar pasteles simultáneamente en una misma caja.*

Implementar utilizando los mecanismos de sincronización que crea mas conveniente