# Programación Concurrente

## Primer Parcial

1. [**Exclusión Mutua**] Considere la siguiente extensión del algoritmo de Peterson para $n$ procesos (con $n > 2$) que utiliza las siguientes variables compartidas:

```
global boolean[] flags = replicate(n,false);
global int turno = 0;
```

Además cada thread está identificado por el valor de la variable local `threadId` (que toma valores entre 0 y $n-1$). Cada thread utiliza el siguiente protocolo.

```
// Seccion no critica
otro = (threadId+1) % n;
flags[threadId] = true;
turno = (turno+1) % n;
while (flag[otro] && turno != threadId);
// Seccion critica
flags[threadId] = false;
// Seccion no critica
```

Mostrar que esta variación no resuelve el problema de la exclusión mutua para $n$ procesos. Indique claramente cuál/es condición/es son violadas.

2. [**Semáforos**] Recientemente en supermercados y otras tiendas están cambiando el sistema por el que los clientes hacen cola para pasar a las cajas. Típicamente, se tiene una cola por caja, un cliente que llega a la línea de cajas escoge una, y espera a los clientes que tenga delante en esa cola. Últimamente se ha pasado al sistema de cola única. En este esquema, todos los clientes hacen una sola cola, y cuando un cajero se libera llama al próximo. Nos interesa comparar esos sistemas implementándolos con semáforos.

   *a*) Implemente un sistema con 10 cajas donde cada caja tiene su propia fila. Escriba un Thread cliente que tome por parámetro la caja donde quiere que le cobren, y un Thread cajero por cada caja que atienda suscesivamente a cada cliente que se encuentre en la cola de esa caja. Atender a un cliente toma una cantidad de tiempo no despreciable.

   *b*) Extienda la solución anterior considerando que cada caja cuenta con varios cajeros que se van turnando para trabajar en ellas. Un cajero tiene una caja asignada y una cantidad de personas que atiende por turno. El cajero debe sentarse en una caja, atender esa cantidad de clientes, y luego liberar la caja. Al contrario de las filas de los clientes, los cajeros que están esperando su turno en la caja no tienen un orden, y cualquiera de los que está esperando puede comenzar a atender.

   *c*) Implemente un sistema con 10 cajas pero donde hay una fila única. Nuevamente hay un Thread Cliente que debe esperar en esta fila hasta ser atendido, y ahora los cajeros deben hacer pasar a los clientes de esta fila. Para este ítem cada caja posee un único cajero. (**Pista:** El cliente al que se le acaba de avisar que una caja se liberó tiene que tener algún lugar para fijarse cuál es la caja que está disponible)