SISTEMAS CONCURRENTES Y DISTRIBUIDOS
Ejercicios adicionales de la práctica 1.

Ejercicio 1. (prod/cons con impresora)

Copia tu solución FIFO al problema del productor-consumidor de la práctica 1 en un archivo llamado **p1_pc_impr.cpp**, y extiéndelo para cumplir estos requisitos adicionales a los del problema original (usando semáforos)

(1) Se debe crear una nueva hebra llamada *impresora*, que ejecuta un bucle finito. En cada iteración debe bloquearse hasta que sea desbloqueada por otra hebra, y después, tras ser desbloqueada, debe imprimir por pantalla el número de elementos que hay en el vector en ese momento (es decir, elementos producidos y añadidos al vector pero todavía no consumidos).

(2) Cuando el productor produzca un número múltiplo de 5, después de añadir dicho número al vector, debe desbloquear a la hebra impresora y luego debe bloquearse.

(3) La hebra impresora, una vez que haya escrito el mensaje por pantalla, desbloqueará al productor y volverá al principio de su ciclo. El número de iteraciones de la hebra impresora es igual al número de múltiplos de 5 que hay entre 0 y N-1 (ambos incluidos), es decir, será N/5 (división entera), donde N es e número total de valores producidos.

Ejercicio 2 (gasolinera)

Implementar un programa con semáforos (en un archivo llamado **p1_gasolinera.cpp**), con los siguientes requerimientos:

(1) El programa está formado por 10 hebras que ejecutan un bucle infinito y que representan a coches que necesitan entrar a repostar a una gasolinera un vez en cada iteración de su bucle. Hay dos tipos de hebras coche: 4 hebras de tipo *gasoil* y 6 hebras de tipo *gasolina*.

(2) Para entrar a la gasolinera un coche debe esperar a que quede libre algún surtidor del tipo que necesita. La gasolinera tiene 3 surtidores de gasolina y 2 para gasoil, inicialmente todos libres. Cada tipo de hebra ejecuta una función del programa distinta (**funcion_hebra_gasolina** y **funcion_hebra_gasoil**)

(3) El tiempo de repostaje y la parte de código de los coches que no están en la gasolinera se simulan con retardos aleatorios. Cada hebra coche debe imprimir un mensaje cuando comienza a repostar y otro cuando termina de repostar (los mensajes y el retraso se deben de incluir en una función llamada **repostar**). Es necesario contabilizar en una variable compartida el número total de surtidores en uso en cada momento.

El pseudo-código de cada tipo de hebra puede ser como se indica a continuación:

```
Procedure HebraTipoX[ n : 0..k ]
begin
  while true do begin
    { esperar hasta que haya un surtidor adecuado libre }
    { incrementar el número de surtidores en uso }
    repostar( n )
    { decrementar el número de surtidores en uso }
    { señalar que se ha dejado el surtidor libre }
    { retraso de duración aleatoria (fuera de gasolinera) }
  end
end
```