**I. Ejercicios UD 2. PROGRAMACIÓN DE HILOS**

1. Diseña un programa en el que dos threads incrementen de manera alternada una variable compartida. La variable solo puede ser incrementada si su valor actual es par o impar, según el thread que intente realizar la operación. Para ello, se tendrán en cuenta los siguientes requisitos:

* La variable inicial será 0.
* El thread 1 incrementará la variable solo si es par.
* El thread 2 incrementará la variable solo si es impar.
* Cada thread imprimirá un mensaje indicando el valor actual y quién lo incrementó.
* El programa finalizará cuando la variable alcance un valor determinado (por ejemplo, 20).

1. Implementa un sistema en el que un productor incrementa un contador y un consumidor lo decrementa. El productor no puede incrementar el contador si este ha alcanzado un límite superior, y el consumidor no puede decrementarlo si ya está en su valor mínimo.

* El productor incrementará el contador hasta un valor máximo (por ejemplo, 10).
* El consumidor decrementará el contador hasta un valor mínimo (por ejemplo, 0).
* Ambos threads deben trabajar coordinados, y no deben modificar el contador simultáneamente.
* El programa imprimirá el valor del contador y la acción realizada (incremento o decremento) en cada operación.

1. Tres threads deben imprimir números en orden ascendente. Cada thread puede imprimir un número solo si es su turno. Para su realización, se tendrán en cuenta los siguientes requisitos:

* Thread 1 imprimirá los números que son múltiplos de 3.
* Thread 2 imprimirá los números que son múltiplos de 3 más 1.
* Thread 3 imprimirá los números que son múltiplos de 3 más 2.
* El programa debe finalizar después de imprimir una cantidad determinada de números (por ejemplo, 30).

1. Simula un sistema en el que varios threads intentan acceder a un conjunto limitado de recursos. Si un recurso no está disponible, el thread debe esperar. Debe cumplir los siguientes requisitos:

* Define un número fijo de recursos disponibles (por ejemplo, 3).
* Cada thread intentará acceder a un recurso, lo usará durante 2 segundos (Thread.sleep(2000)) y luego lo liberará.
* Debe asegurarse que nunca haya más threads usando recursos que la cantidad disponible.
* Al finalizar, cada thread imprimirá cuántas veces utilizó un recurso.