Tecnología y paradigmas de programación. Laboratorio 1. Grupos 1, 2 y 3.

TPL y PLINQ. En algún momento puede que usemos este fragmento de código que me ha suministrado amablemente vuestro profesor de teoría. Si se incluye en una lambda que se pase a Parallel.ForEach o a Parallel.For permite averiguar cuantos threads se usan. Evidentemente, antes hay que declarar la lista `threadIds`, es una lista de enteros.

```
 if (!threadIds.Contains(Thread.CurrentThread.ManagedThreadId))
    {
    Console.WriteLine("Thread ID = " +
Thread.CurrentThread.ManagedThreadId);
    threadIds.Add(Thread.CurrentThread.ManagedThreadId);
    }
```

Ejercicio 1. En este ejercicio se va a tomar como punto de partida el proyecto vector.modulus de la solución PLINQ, incluida en los ejemplos de teoría. Estudiar el código, en el fuente se definen varios métodos y se realizan comparativas entre implementaciones secuenciales y paralelas.

Tal y como se ha hecho en alguna de las tareas autónomas, es posible usar un diccionario para contabilizar cuantas veces se repite cada valor en una colección. Implementar dos métodos que calculen cuantas veces se repite cada entero en un array de este tipo, uno usando `Parallel.For` y otro usando `Parallel.ForEach`.

Comprobar el resultado en modo release, varias veces, usando distintos tamaños de array, alguna vez de un buen número de elementos. El número de veces que aparece cada número debe de ser el mismo que si se calcula mediante un algoritmo serie y no deben de lanzarse excepciones. Si no es así, comprobar que la suma de las frecuencias absolutas (la suma de los valores del histograma) es igual al tamaño de la muestra. Si no es el mismo e incluso cambia entre ejecuciones, intentar averiguar la razón y corregir el posible fallo.

Puede que tras corregir el fallo el algoritmo paralelo no sea más rápido que el serie, probar otra alternativa: calcular varios histogramas particionando los datos, por ejemplo para las posiciones pares e impares del array, lanzar con `Invoke`. Acumular los histogramas obtenidos: cada elemento del histograma total es la suma de los elementos correspondientes de los histogramas parciales.

Ejercicio 2. Existen casos en los que es posible paralelizar tareas y casos en los que no. Este método

```
static double varLinq(IEnumerable<short> vector)
    {
    var mean= vector.Aggregate(0.0,(acc,el) => acc+el) /
vector.Count();
    return vector.Aggregate(0.0, (acc, el) => acc + Math.Pow((el -
mean),2.0))/vector.Count();
    }
```

calcula la varianza de un vector de short usando la fórmula E((x-E(x))^2), E es la media y x representa la muestra, es por lo tanto la media del cuadrado de la diferencia entre cada elemento y la media de todos los elementos: se necesita conocer en primer lugar E(x). Comprobar si existe alguna ganancia en eficiencia usando PLINQ.

Estos métodos calculan E(x) y E(x^2):

```
static double mean(IEnumerable<int> vector)
    {
    return vector.Aggregate(0.0, (acc, el) => acc + el) / (double)
vector.Count();
    }
static double meanX2(IEnumerable<int> vector)
    {
    return vector.Aggregate(0.0, (acc, el) => acc + el*el) /
(double)vector.Count();
    }
```

En este caso es posible usarlos para implementar a su vez un método que los invoque en paralelo (usando `Parallel.Invoke`) para calcular esas magnitudes para un parámetro de tipo short[]. Al terminar la invocación, calcular su varianza usando la fórmula E(x^2) - (E(x))^2. Comprobar si en este caso existe alguna ganancia en eficiencia.

Ejercicio 4. Generar una colección con enteros aleatorios ordenados de menor a mayor (por ejemplo, 100 números entre 2 y 10000), producir una lista con los valores de esa colección que sean primos. Implementar de tres formas distintas, usando `Parallel.For`, `Parallel.ForEach` y PLINQ. ¿Alguna particularidad en como se almacenan/producen los resultados?

Ejercicio 5. Calcular el módulo de un vector usando `Parallel.For` o `Parallel.ForEach` ¿es necesario usar algo más?.

Ejercicio 6. Sumar vectorialmente dos vectores, escoger la alternativa más adecuada (`Parallel.For`, `Parallel.ForEach`, PLINQ).