Compactando secuencias

En la clase `ExamenFinal.Iteradores` implemente el método `AgregaPorBloques<T>`, el cual recibe entre sus parámetros un `IEnumerable<T>` y devuelve un `IEnumerable<T>` formado de aplicar a los bloques de una cantidad `longitud` de elementos consecutivos la operación de agregación dada por el delegado `operacion`.

```
namespace Weboo.Utils
{
    public delegate T Operacion<T>(T a, T b);
}
…
namespace ExamenFinal
{
    public class Iteradores
    {
        public static IEnumerable<T> AgregaPorBloques<T>
               (IEnumerable<T> elementos, Operacion<T> operacion, int longitud)
        {
            //TODO: Escriba su código aquí
        }
    }
}
```

Por ejemplo, si se desea sumar los números del 1 al 10 de tres en tres, para el `IEnumerable<int> {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}` el resultado sería el `IEnumerable<int>` que devuelva `{6, 15, 24, 10}` obtenido al realizar las operaciones 1+2+3, 4+5+6, 7+8+9, 10. Note que como en el último grupo no alcanzan a haber tres consecutivos sino solo el valor 10 el resultado de la suma es 10. El siguiente ejemplo muestra cómo utilizar el método para esta tarea:

```
    class Program
    {

        static int Suma(int a, int b)
        {
            return a + b;
        }

        static void Main(string[] args)
        {
            int[] valores = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
            IEnumerable<int> resultados = Iteradores.AgregaPorBloques(valores, Suma, 3);
            foreach (int i in resultados)
                Console.WriteLine("{0} ", i);
        }
    }
```

Por otra parte, si el objetivo es obtener los menores elementos de dos en dos del `IEnumerable<string> {“rojo”, “verde”, “amarillo”, “azul”, “negro”, “blanco”}`, el resultado sería un `IEnumerable<string>` con `{“rojo”, “amarillo”, “blanco”}` y se puede proceder tal como se ilustra a continuación:

```
static T Min<T>(T a, T b) where T : IComparable<T>
{
        if (a.CompareTo(b) > 0)
                return b;
        return a;
}

static void Main(string[] args)
{
        string[] colores = { "rojo", "verde", "amarillo", "azul", "negro", "blanco" };
        IEnumerable<string> resultados = Iteradores.AgregaPorBloques(colores, Min, 2);
        foreach (string r in resultados)
                Console.WriteLine(r);
}
```

En este caso se realizaron las comparaciones “rojo” con “verde”, “amarillo” con “azul” y “negro” con “blanco”, obteniéndose “rojo”, “amarillo” y “blanco” respectivamente.

NOTA: Puede considerar que el `IEnumerable<T> elementos` no será nunca `null`. Si elementos es una secuencia vacía entonces su función deberá dar como resultado un `IEnumerable<T>` también vacío. Implemente todo el ejercicio en la plantilla de código, adjunta a esta orientación.