Appello TAP del 11/07/2017

Scrivere nome, cognome e matricola sul foglio protocollo, indicando anche se avete nel piano di studi TAP da 6 CFU (quello attuale) o da 8 CFU (quello "vecchio"). Avete a disposizione due ore.

Esercizio 1 (10 punti)

Scrivere l'extension-method `GroupMin` che traduce sequenze di elementi di tipo `int` in sequenze di elementi di tipo `int`. Il metodo `GroupMin` invocato su `individuals` e un intero `groupSize` segmenta `individuals` in gruppi di `groupSize` elementi consecutivi e restituisce il minimo fra gli elementi di ciascun gruppo.

Per esempio, il seguente frammento di codice

```
foreach (var d in new[] {3, 3, 6, 4, -3, 6}.GroupMin(3))
                Console.Write("{0} ", d);
Console.WriteLine();
```

stampa:

```
3 -3
```

(perché 3 è il minimo di $\{3, 3, 6\}$ e -3 è il minimo di $\{4, -3, 6\}$)

Il metodo dovrà prendere come parametro "this" `individuals`, la sequenza sorgente. Nota: la sequenza può anche essere infinita.

Il metodo deve sollevare l'eccezione...

- `ArgumentNullException` se `individuals` è `null`;
- `ArgumentOutOfRangeException` se `groupSize` non è strettamente positivo
- `ArgumentException` se `individuals` rappresenta una sequenza finita di lunghezza *non* multipla di `groupSize`

Esercizio 2 ([3+3+4] = 10 punti)

Implementare, usando NUnit ed eventualmente Moq, i seguenti test relativi al metodo `GroupMin`, dell'esercizio precedente.

1. Test parametrico con parametri interi `size` e `groupNumber`.

   Input della chiamata sotto test: `individuals` deve essere la sequenza dei primi `groupNumber` $*$`size` interi, partendo da 0 e `groupSize` deve essere `size`.

   Output atteso: per calcolare il risultato si consideri che il gruppo i-esimo inizierà con il valore $i*$`size` e terminerà con $(i+1)*$`size` -1; ad esempio se `size` $= 2$ e `groupNumber` $= 3$, allora i gruppi saranno $\{0, 1\}, \{2, 3\}, \{4, 5\}$.

2. Input della chiamata sotto test: `individuals` deve essere la sequenza dei numeri interi, partendo da 100 e `groupSize` deve essere -1.

   Output atteso: deve essere sollevata un'eccezione di tipo `ArgumentOutOfRangeException`.

3. Input della chiamata sotto test: `individuals` deve essere la sequenza dei numeri pari, partendo da 42 e `groupSize` deve essere 5.

   Il test deve verificare la correttezza dei primi 666 elementi del risultato (calcolati analogamente a quanto suggerito al primo punto).

Esercizio 3 (10 punti)

Applicare i principi della dependency injection per fare refactoring della seguente classe `C` eliminando le dipendenze indesiderate. Introdurre i tipi necessari e modificare `C` di conseguenza.

Dire se è necessario modificare altri tipi o introdurne di nuovi e, in caso positivo, descrivere le modifiche necessarie.

```
public class C
    {
        public E MyE { get; private set; }
        public D[] MyArray { get; private set; }

        public C()
        {
            this.MyArray = new D[42];
            for (int i=0;i<42;++i){MyArray[i] = new D();}
            this.MyE = new E();
        }

        public string M(bool x, int y)
        {
            return this.MyE.H( this.MyArray[y], x);
        }
    }
```