Appello TAP del 28/01/2013

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

Chi deve sostenere TAP da 6 CFU dovrà svolgere solo i primi tre esercizi; per loro il punteggio indicato nel testo sarà scalato, di conseguenza, di $\frac{\sum_{i=1}^{4} PuntiEs_i}{\sum_{i=1}^{3} PuntiEs_i}$

Avete a disposizione mezzora per esercizio (quindi, un'ora e mezza per chi deve sostenere TAP da 6 CFU e due ore per TAP da 8 CFU).

Esercizio 1 (10 punti)

Scrivere l'extension-method generico `Expand<T>` che, presa una sequenza di elementi di tipo `T`, restituisce una nuova sequenza dove tutte le occorrenze di un particolare valore (di tipo `T`) sono state sostituite da una specifica sequenza di valori (di tipo `T`). Il metodo dovrà prendere come parametri:

1. (come parametro "this") `sequence`, la sequenza sorgente. Nota: questa sequenza può anche essere infinita;
2. `value`, il valore da sostituire;
3. `newValues`, un array di T, contenente i valori da sostituire al posto di `value`.

Per esempio, il seguente frammento di codice scrive sullo standard output i numeri: 1, 7, 8, 9, 1, 7, 8, 9, 3.

```
new [] {1, 2, 1, 2, 3}
    .Expand(2, new [] {7, 8, 9})
    .ToList()
    .ForEach(Console.WriteLine);
```

Il metodo deve sollevare l'eccezione...

- `ArgumentNullException` se `sequence` o `newValues` sono null
- `ArgumentException` se l'array `newValues` contiene meno di due elementi

Esercizio 2 (3+3+3 = 9 punti)

- Elencare, descrivendoli a parole, una lista di test significativi per il metodo `Expand<T>`, dell'esercizio precedente.
- Implementare, usando NUnit, due test della lista precedente; uno che vada a testare un caso "buono" (ovvero, dove ci si aspetta che l'invocazione di `Expand` vada a buon fine) e uno che vada a testare un caso "cattivo" (ovvero, dove ci si aspetta che l'invocazione di `Expand` sollevi un'eccezione).
- Il code-coverage dei due test implementati è il 100% rispetto alla vostra implementazione dell'Esercizio 1? Motivare la risposta.

Esercizio 3 (6 punti)

Applicando i principi della dependency injection, eliminare dalla seguente classe `C` le dipendenze da `D` ed `E`. Introdurre i tipi necessari e modicare `C` di conseguenza. Dire se è necessario modificare anche `D` ed `E`; in caso positivo, descrivere le modifiche necessarie.

```
public class C {
    private double _k;
    private D _d;
    private E _e;

    public C(double k) {
        this._k = k;
        this._d = new D();
        this._e = new E();
    }

    public double M(int x, int y) {
        if (x > y)
            return this._d.F(x);
        if (x == y)
            return this._d.G(y);
        return this._e.H(this._k, x, y);
    }
}
```

Esercizio 4 (5 punti)

Scrivere un metodo che, preso un array di stringhe `a` e un intero `i`, restuisca il numero di stringhe, all'interno dell'array `a`, più lunghe di `i` caratteri. Dare un'implementazione che sfrutti eventuale hardware multicore.

Sollevare opportune eccezioni (standard o definite da voi) dove ritenuto necessario.