Appello TAP del 09/09/2013

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 non dovrà svolgere un esercizio (indicato nel seguito); per loro il punteggio indicato nel testo sarà scalato, di conseguenza.

Esercizio 1 (15 punti)

Scrivere l'extension-method generico `MergeWith<T1,T2,TResult>` che, invocato su:

- una sequenza $s_1 = a_1, a_2, \ldots$ (di elementi di tipo `T1`), con parametri
- un'altra sequenza $s_2 = b_1, b_2, \ldots$ (di elementi di tipo `T2`) e
- una funzione f che, presi in input un parametro di tipo `T1` e uno di tipo `T2`, restituisce un elemento di tipo `TResult`

restituisce la sequenza $f(a_1, b_1), f(a_2, b_2), \ldots$ (di elementi di tipo `TResult`).

La lunghezza della sequenza di output dovrà corrispondere alla lunghezza della sequenza più corta fra s_1 e s_2; in altre parole, il metodo termina il suo lavoro quando una delle due sequenze finisce.

Nota bene: il metodo deve gestire anche sequenze infinite.

Sollevare delle eccezioni, standard o definite da voi, ove ritenuto necessario.

Esempio di uso:

```
public static IEnumerable<T> InfSeq<T>(T x) {
    for (;;)
        yield return x;
}
// ...
foreach (var x in new[] {1,2,3}.MergeWith(new[] {4,5,6}, (a, b) => a + b))
    Console.WriteLine(x); // writes: 5,7,9
foreach (var x in new[] {1}.MergeWith(new[] {4,5,6}, (a, b) => a + b))
    Console.WriteLine(x); // writes: 5
foreach (var x in new[] {1,2}.MergeWith(new[] {7}, (a, b) => a * b))
    Console.WriteLine(x); // writes: 7
foreach (var x in InfSeq("a").MergeWith(InfSeq(2), (a, b) =>a+b).Take(3))
    Console.WriteLine(x); // writes: a2,a2,a2
foreach (var x in InfSeq(6).MergeWith(InfSeq(7), (a,b)=>a*b))
    Console.WriteLine(x); // writes: 42,42,42,42,42,...
```

Esercizio 2 (10 punti)

- Implementare, usando NUnit, una serie di test per verificare il metodo `MergeWith` (dell'esercizio 1).
- Dare, se possibile, un'implementazione *sbagliata* del metodo che passi tutti i vostri test, oppure spiegare perché non è possibile.

Esercizio 2 (5 punti) — solo per chi ha TAP da 8 CFU

Considerate i seguenti metodi (appartenenti a una qualche classe, non rilevante ai fini dell'esercizio):

```
private int M1(bool b) {/*...*/}
private int M2(double d) {/*...*/}
private double M3(int x, int y) {/*...*/}

public double M(bool b, char c) {
     return M3(M1(!b), M2(Math.PI));
}
```

sapendo che `M1` e `M2` potrebbero essere eseguiti in parallelo, scrivere le seguenti varianti di `M`:

- `OptimizedM`, che (potenzialmente) esegue `M1` ed `M2` in parallelo, pur rimanendo sincrono
- `MAsync1`, una versione asincrona di `M` che (potenzialmente) esegue `M1` ed `M2` in parallelo, senza utilizzare `async`/`await` di C# 5
- `MAsync2`, una versione asincrona di `M` che (potenzialmente) esegue `M1` ed `M2` in parallelo, utilizzando `async`/`await` di C# 5