Appello TAP del 05/06/2013

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 l'ultimo esercizio; per loro il punteggio indicato nel testo sarà scalato, di conseguenza, di $\frac{\sum_{i=1}^{n} PuntiEs_i}{\sum_{i=1}^{n-1} PuntiEs_i}$, con n il numero totale di esercizi.

Tempo a disposizione: *un'ora e mezza* per chi deve sostenere TAP da *6 CFU*, *due ore* per TAP da *8 CFU*.

Esercizio 1 (12 punti)

Scrivere il metodo generico `Classify<T>` che, presa una sequenza s (di elementi di tipo `T`) e un numero arbitrario di predicati $p_1, \ldots, p_n$, restituisce un array di $n+1$ liste $[l_1, \ldots, l_{n+1}]$ tali che:

- l_1 contenga gli elementi di s che soddisfano p_1
- l_2 contenga gli elementi di s che non soddisfano p_1, ma soddisfano p_2
- l_3 contenga gli elementi di s che non soddisfano p_1 e p_2, ma soddisfano p_3
- ...
- in generale, per $i \in \{1, \ldots, n\}$, l_i contenga gli elementi di s che non soddisfano nessuno dei predicati $p_1, \ldots, p_{i-1}$, ma soddisfano p_i
- infine, l_{n+1} contenga gli elementi di s che non soddisfano nessuno dei predicati $p_1, \ldots, p_n$

Il metodo non deve alterare l'ordine degli elementi, ma solo "smistarli" nelle varie liste $l_1, \ldots, l_{n+1}$. In altre parole, se $l_i = \ldots a \ldots b \ldots$, allora anche s può essere scritta come $s = \ldots a \ldots b \ldots$ (dove, naturalmente, i puntini "..." indicano sequenze arbitrarie, anche vuote). Sollevare delle eccezioni, standard o definite da voi, ove ritenuto necessario.

Esempio di uso:

```
var numbers = Enumerable.Range(-3, 10); // -3, -2, ..., 6
var partition = Classify(
    numbers,
    i => i >= 0 && i % 2 == 0,
    i => i >= 0 && i % 2 != 0,
    i => i < -2);
/* partition[0]=0 2 4 6
 * partition[1]=1 3 5
 * partition[2]=-3
 * partition[3]=-2 -1
 */
```

Esercizio 2 (6+3+4 = 13 punti)

In questo esercizio vogliamo testare il metodo `Classify<T>`, dell'esercizio precedente.

- Implementare, usando NUnit, uno o più test per convincersi che il metodo usi tutti i predicati che gli vengono passati (in almeno qualche caso significativo).
- Elencare, descrivendoli a parole, una lista di altri test che ritenete significativi.
- Implementare, usando NUnit, quattro test della lista precedente: tre che vadano a testare un caso "buono" (ovvero, dove ci si aspetta che l'invocazione di `Classify` vada a buon fine) e uno che vada a testare un caso "cattivo" (ovvero, dove ci si aspetta che l'invocazione di `Classify` sollevi un'eccezione).

Esercizio 3 (2+2+1=5 punti)

Supponete di avere il seguente metodo:

```
string Foo(string [] bar, int baz) { /* ... */ }
```

(quello che fa è irrilevante ai fini dell'esercizio)

1. Come cambia la segnatura del metodo `Foo` se vogliamo renderlo asincrono seguendo i pattern...
   - Asynchronous Programming Model (APM)?
   - Task-based Asynchronous Programming (TAP)?
2. Implementare la versione asincrona che segue il pattern TAP. Nota: si vuole solo rendere il metodo asincrono, non è richiesta nessuna ottimizzazione particolare.
3. Dare un esempio d'uso (=invocazione ed estrazione del risultato) della vostra implementazione