Appello TAP del 7/09/2015

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") e se avete consegnato i test durante l'anno e intendete usufruire del bonus così conseguito.

Chi deve sostenere TAP da 6 CFU non deve svolgere l'esercizio 1 e chi ha consegnato i test durante l'anno (e intende usufruire del bonus conseguito) non deve svolgere l'esercizio 4; per loro il punteggio indicato nel testo sarà scalato, di conseguenza, in modo che il massimo conseguibile sia sempre 30. Avete a disposizione mezzora per esercizio. In sintesi:

Tipo TAP	Bonus Test	Esercizi da svolgere	Tempo a disposizione	Fattore di normalizzazione
6CFU	sì	2 e 3	1h	$\frac{30}{10+9}$
6CFU	no	2, 3 e 4	1h 30'	$\frac{30}{10+9+6}$
8CFU	sì	1, 2 e 3	1h 30'	$\frac{30}{5+10+9}$
8CFU	no	tutti (1, 2, 3 e 4)	2h	1

Esercizio 1 (punti 5)

- Dare l'implementazione del metodo asincrono `SelectChar` che, presi in input un array `indexes` di `Task<int>` ed un secondo array `words` di `Task<string>` di pari lunghezza, restuisca un `Task<string>`. La stringa calcolata da questo task avrà come carattere i-esimo il carattere `words[i].Result[indexes[i].Result]`.

  Ad esempio, sugli array `words` == [`t0`,`t1`,`t2`] e `indexes` == [`tt0`,`tt1`,`tt2`] il task risultato di `SelectChar` calcolerà `t0.Result[tt0.Result]`, `t1.Result[tt1.Result]`, `t2.Result[tt2.Result]`

  Si sollevino opportune eccezioni per segnalare i casi di errore.

- Dare un esempio di invocazione del metodo implementato nel punto precedente.

Esercizio 2 (punti 3+10 = 10) Scrivere l'extension-method generico `MacroExpansion<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`, una sequenza di T, contenente i valori da sostituire al posto di `value`. Nota: anche questa sequenza può anche essere infinita.

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

Il metodo deve sollevare l'eccezione `ArgumentNullException` se `sequence` o `newValues` sono null.

Esercizio 3 (punti 3 + 6 = 9)

- Definire un custom-attribute `ManHourAttribute` che permetta di specificare, solo su metodi, quante ore sono state dedicate alla sua implementazione da parte di uno specifico programmatore, eventualmente usando annotazioni multiple su uno stesso metodo per diversi autori. Esempio d'uso:

```
[ManHour("pippo",3)]
[ManHour("pluto",5)]
public void M() { /* ... */ }
```

- Utilizzare il custom-attribute definito al punto precedente per definire un metodo che, preso in input il nome di un programmatore e una classe, calcoli quante ore quel programmatore ha dedicato a quella classe (si ignorino ventuali metodi non pubblici).

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

- Elencare, descrivendoli a parole, una lista di test significativi per il metodo `MacroExpansion<T>`, dell'esercizio 2.
- 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 `MacroExpansion<T>` vada a buon fine) e uno che vada a testare un caso "cattivo" (ovvero, dove ci si aspetta che l'invocazione di `MacroExpansion<T>` sollevi un'eccezione).
- Implementare, usando NUnit, un test in cui `MacroExpansion<T>` venga invocato su (almeno) un argomento "infinito".