Appello TAP del 26/06/2014

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 gli ultimi tre esercizi; per loro il punteggio indicato nel testo sarà scalato, di conseguenza, di $\frac{\sum_{i=1}^{4} PuntiEs_i}{\sum_{i=2}^{4} PuntiEs_i}$ 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 (5 punti)

Supponete di avere un'entità `Boh`, con un metodo (di istanza) `m()` senza parametri che restituisce un intero, e una classe `MyContext: DbContext` con, fra le altre, una proprietà `DbSet<Boh> Bohs` che restituisce tutte le entità di quel tipo, come d'uso, e un costruttore senza parametri.

Definire un metodo che, preso in input un predicato p su `Boh`, produce la somma dell'esecuzione di `m()` su tutte le entità che soddisfano p, sfruttando per quanto possibile eventuale hardware multicore, assumendo che tutti i metodi e predicati descritti possano essere eseguiti in parallelo.

Esercizio 2 (9 punti)

Scrivere l'extension-method `Initials` che, presa una sequenza `names` di stringhe corrispondenti a nomi in alfabeto latino, restituisce la sequenza delle iniziali corrispondenti maiuscole, sollevando opportune eccezioni nel caso un elemento di `names` non sia un nome, ovvero non sia la concatenazione di stringhe alfabetiche separate da uno o più spazi. Per esempio, il seguente frammento di codice

```
var names = new string[] { "paolino paperino", "Daisy      Duck", "Paperon de paperoni" }
foreach (var ii in names.Initials())
    Console.WriteLine(ii);
```

stampa, nell'ordine, `PP`, `DD`, `PDP`.

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

Il metodo deve sollevare l'eccezione...

- `ArgumentNullException` se `names` o uno dei suoi elementi è `null`;
- `ArgumentOutOfRangeException` se uno degli elementi di `names` contiene caratteri che non sono lettere o spazi.

A seconda della soluzione che volete adottare, potreste trovare utili alcuni dei seguenti metodi:

```
Regex.IsMatch(string input, string pattern)
Char.IsLetter(Char)
ToUpper() // instance method on char

// instance methods on string:
public string[] Split(params char[] separator)
public string[] Split(char[] separator, StringSplitOptions options)
    //public enum StringSplitOptions contains None and RemoveEmptyEntries
```

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

- Elencare, descrivendoli a parole, i test significativi per il metodo `Initials`, dell'esercizio precedente.
- Implementare, usando NUnit ed eventualmente Moq, due test della lista precedente; uno che vada a testare un caso "buono" (ovvero, dove ci si aspetta che l'invocazione di `Initials` vada a buon fine) e uno che vada a testare un caso "cattivo" (ovvero, dove ci si aspetta che l'invocazione di `Initials` sollevi un'eccezione).
- Implementare, usando NUnit ed eventualmente Moq, un test significativo in cui `Initials` è instanziato su una sequenza infinita.

Esercizio 4 (7 punti)

Utilizzando un meccanismo di comunicazione basato sugli eventi, implementare le parti essenziali di un sistema (futuribile) che emette automaticamente le multe per eccesso di velocità.

Si assumano date le seguenti classi, che possono essere modificate a piacere ("..." indica le parti non rilevanti ai fini dell'esercizio), in modo da ottenere che quando la velocità di una macchina cambia, se la nuova velocità è superiore al limite previsto in quel punto venga invocato il metodo `IssueSpeedTicket` con i corretti parametri.

```
public class Person {/*...*/}

public class GPSLocation
{
    /*...*/
    public int SpeedLimit { get; private set; }
}

public class Car
{
    /*...*/
    public Person Owner { get; set; }
    public GPSLocation Location { get; set; }
    public int Speed { get; set; }
}

public class Police
{
    /*...*/
    public void IssueSpeedTicket(Car culprit, DateTime when, int speed) { /*...*/}
    public void RegisterCar(Car car) { /*...*/}
}
```