Appello TAP del 13/02/2013

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 (9 punti)

Scrivere il metodo `RandomSample<T>` che, presi:

- una sequenza finita $a_1, \ldots, a_n$ di elementi di tipo `T`,
- un numero intero non-negativo k e
- un flag `noDup`,

restituisca una nuova sequenza di k elementi presi a caso dalla sequenza di input, ovvero una sequenza $a_{i_1}, \ldots, a_{i_k}$ dove gli indici i_j sono numeri interi casuali nel range $[1, n]$. Inoltre, se `noDup`, $s \neq r \Rightarrow i_s \neq i_r$, ovvero, un elemento della sequenza di input non può esser scelto due volte (si noti che la sequenza di output potrebbe comunque contenere degli elementi duplicati, se la sequenza di input contiene elementi duplicati).

Per generare numeri casuali potete usare la classe `Random`, istanziandola usando il costruttore di default, e il suo metodo (d'istanza) `Next`(int) che (da MSDN) "Returns a nonnegative random number less than the specified maximum".

Il metodo `RandomSample<T>` deve sollevare:

- `ArgumentNullException` se la sequenza è null
- `ArgumentException` se l'intero k è negativo, oppure $k > 0$ e la sequenza è vuota
- `InvalidOperationException` se `noDup` e $k > n$

Esercizio 2 (9 punti)

La specifica del metodo d'istanza void `ForEach(Action<T> action)` della classe `List<T>` è:

```
Performs the specified action on each element of the List<T>.

Exceptions:
ArgumentNullException when action is null.

Remarks:
The Action<T> is a delegate to a method that performs an action
on the object passed to it. The elements of the current List<T>
are individually passed to the Action<T> delegate.

This method is an O(n) operation, where n is Count.

Modifying the underlying collection in the body of the Action<T>
delegate is not supported and causes undefined behavior.
```

Utilizzando NUnit e, se necessario, Moq, scrivere degli unit-test per verificare che il metodo `List<T>.ForEach` rispetti la sua specifica.

Esercizio 3 (5 punti)

Definire un custom-attribute `MyFactoryAttribute` che permetta di specificare, solo su classi e interfacce, qual è il tipo della propria factory. Esempio d'uso:

```
[MyFactory(typeof(FooFactory))]
interface IFoo { /* ... */ }
```

Esercizio 4 (7 punti)

Si consideri la seguente dichiarazione di metodo:

```
T M<T>(Func<T> f1, Func<T> f2, Func<T, T, T> f3) {
    if (f1 == null) throw new ArgumentNullException("f1");
    if (f2 == null) throw new ArgumentNullException("f2");
    if (f3 == null) throw new ArgumentNullException("f3");
    return f3(f1(), f2());
}
```

Dare, seguendo il pattern TAP (Task-based Asynchronous Programming), la versione asincrona di `M<T>`:

1. Senza nessuna assunzione particolare su `f1` ed `f2`
2. Assumendo che `f1` ed `f2` possano essere eseguite in parallelo