Appello TAP del 4/2/2022

Scrivere nome, cognome e matricola sul foglio protocollo. Avete a disposizione due ore e mezza.

Esercizio 1 (9 punti)

Scrivere l'extension-method `GetContainedNumbers` che, invocato su `source`, una sequenza eventualmente nulla di stringhe non nulle, restituisce la sequenza (non nulla) di interi in cui ciascun elemento è ottenuto leggendo le sole cifre contenute nella stringa nella stessa posizione della sequenza sorgente, ignorando tutti gli altri caratteri e interpretando la sequenza di cifre come un numero in base 10.

Per esempio, il seguente frammento di codice

```
foreach(var d in
        new[]{"1qui7", "q8u8o", "-1024", "0q0ua0"}.GetContainedNumbers())
    Console.Write("{0}, ",d);
Console.WriteLn();
```

stampa:

```
17, 88, 1024, 0,
```

Il metodo dovrà prendere come parametro "this" `source`, la sequenza sorgente, che può anche essere infinita.

Il metodo deve sollevare l'eccezione...

- `ArgumentNullException` se `source` è `null`;
- `ArgumentException` se in `source` compare una stringa che non contiene nessuna cifra.

La dichiarazione del metodo deve tenere conto degli aspetti di nullabilità dei tipi reference nella scelta dei tipi.

Quando possibile, il metodo dovrà sollevare le eccezioni richieste senza che sia necessario enumerarne il risultato.

Esercizio 2 (7 punti)

Implementare, usando NUnit, i seguenti test relativi a `GetContainedNumbers`, dell'esercizio 1.

1. Input della chiamata sotto test: `source` è la sequenza `"45", "-8gg","lll  lo sembra ma non lo e'"`

   Output atteso: una `ArgumentException`.

2. Input della chiamata sotto test: `source` è la sequenza `"f55h7", "90", "-45", "H 6YY5"`

   Output atteso: la sequenza `557, 90, 45, 65`

3. Input della chiamata sotto test: `source` è la sequenza **infinita** i cui elementi sono la concatenazione del carattere `'a'`, un numero progressivo (partendo da 0) e il carattere `'z'`, cioè `"a0z"`, `"a1z"`, `"a2z"`, `"a3z"`....

   Output atteso: la sequenza dei numeri interi positivi. Essendo un output infinito, il test dovrà verificare solo che i primi 100 valori nel risultato siano i numeri da 0 a 99.

Esercizio 3 (9 punti)

Per ciascuna delle seguenti affermazioni, indicate se è vera o falsa

1. In C#, se un metodo solleva una eccezione di tipo `E`

Vero	Falso	
☐	☒	lo stesso metodo deve anche catturarla e gestirla
☐	☒	nell'intestazione del metodo `E` deve comparire all'interno della clausola `throws`
☐	☒	lo stesso metodo non può sollevare altre eccezioni di tipo `E`

2. Nell'istante in cui un oggetto di tipo `C` con accesso esclusivo a un file diventa irraggiungibile

Vero	Falso	
☐	☒	il file viene immediatamente rilasciato automaticamente
☐	☒	se `C` non implementa `IDisposable` si ha errore dinamico
☒	☐	anche se `C` implementa `IDisposable` il file potrebbe restare "lockato"

3. Se un oggetto di classe `C` ha bisogno di un logger di tipo `L`, secondo la dependency-injection:

Vero	Falso	
☐	☒	deve esistere anche la classe factory per `L` da usare in `C`
☒	☐	il DI container intercetterà `new L()` nei costruttori di `C` e materializzerà il logger
☐	☒	i costruttori di `C` devono avere un parametro di tipo `L`

4. Per poter passare come parametro un metodo, il tipo del parametro può essere

Vero	Falso	
☒	☐	`Func<..>`, `Action<...>` o altro tipo *delegate*
☒	☐	un tipo *lambda*
☐	☒	`FunctionPointer<...>` o altro tipo puntatore

5. Per definire un custom-attribute

Vero	Falso	
☐	☒	bisogna essere in un progetto per la piattaforma .NET Standard Library
☐	☒	si deve avere un riferimento a `System.Annotations.CustomAttributes`
☐	☒	basta estendere la classe `CustomAttribute`

6. Considerando i vari tipi di passaggio di parametri in C# 9

Vero	Falso	
☐	☒	per dichiarare un singolo parametro si possono usare simultaneamente `ref` e `in`
☒	☐	dichiarare un parametro come `out` impone vincoli statici sul corpo del metodo
☒	☐	usare `ref` impone vincoli statici sui parametri attuali usabili nella chiamata

7. Se in uno unit-test compaiono più asserzioni a livello di annidamento top, cioè non contenute in altri statement/espressioni:

Vero	Falso	
☐	☒	il test runner solleva un'eccezione (uno unit test non può contenere più asserzioni)
☒	☐	alla prima asserzione fallita il test termina l'esecuzione
☐	☒	il test runner indica tutte le asserzioni fallite

8. Anche se il server git di riferimento per un repo (l'*origin* del repo) non è raggiungibile, i seguenti comandi possono ugualmente avere successo:

Vero	Falso	
☒	☐	`git commit`
☐	☐	`git push`
☐	☐	`git branch [branch-name]`

9. Confrontando le interfacce `IQueryable` e `IEnumerable` **// IQueryable deriva da IEnumerable**

Vero	Falso	
☒	☐	`IQueryable` ha più membri di `IEnumerable`
☐	☒	`IEnumerable` ha più membri di `IQueryable`
☐	☒	`IEnumerable` ha gli stessi membri di `IQueryable`, ma con diverse implementazioni