TAP: Appello del 6/7/2022

Scrivere nome, cognome e matricola sul foglio protocollo e sul foglio con le risposte al quiz. Avete a disposizione due ore e mezza.

Esercizio 1 (9 punti)

Scrivere un extension method `Repeat <T>` che prenda in input:

- una sequenza non nulla `s`, che potrebbe essere infinita, di tipo `T` come parametro this
- un intero strettamente positivo `multiplicity`
- un elemento `forbidden` di tipo `T` che non deve comparire in `s`

e restituisca una sequenza dove ogni elemento di `s` viene ripetuto `multiplicity` volte. Se in `s` compare `forbidden` il metodo dovrà sollevare `ArgumentException` e se `multiplicity` non è strettamente positivo dovrà sollevare `ArgumentOutOfRangeException`.
Esempi:

```
var a = new[] {1, 3, 7};
// a.Repeat(1, 5) = 1, 3, 7
// a.Repeat(2, -1) = 1, 1, 3, 3, 7, 7
// a.Repeat(3, 10) = 1, 1, 1, 3, 3, 3, 7, 7, 7
```

Mentre ad esempio `a.Repeat(5, 3)` dovrà sollevare `ArgumentException` e `a.Repeat(0, 33)` dovrà sollevare `ArgumentOutOfRangeException`.

Esercizio 2 (7 punti)

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

1. Input della chiamata sotto test: `s` è la sequenza {8, 11, 35}, `multiplicity` è l'intero 2 e `forbidden` è 112.
   Output atteso: la sequenza {8, 8, 11, 11, 35, 35}.
2. Input della chiamata sotto test: `s` è la sequenza {3.8, 24.31, 3.675}, `multiplicity` è l'intero 0 e `forbidden` è −4.67.
   Output atteso: una eccezione di tipo `ArgumentOutOfRangeException` sollevata senza la necessità di enumerare, neppure parzialmente, il risultato.
3. Test parametrico con due parametri, `position` e `badGuy`, di tipo intero. Se `position` non è strettamente positivo il test dovrà risultare *inconclusive*.
   Input della chiamata sotto test:
   - `s` è una sequenza infinita di numeri generati casualmente tutti diversi da `badGuy` tranne in posizione `position` dove si trova proprio `badGuy`
   - `multiplicity` è 3
   - `forbidden` è `badGuy`

   Output atteso: una eccezione di tipo `ArgumentException` sollevata dopo aver correttamente restituito 3*`position` elementi.

Esercizio 3 (9 punti)

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

1. Nell'ambito dei custom attribute

Vero	Falso	
☐	☒	i custom attribute non esistono più nelle versioni più recenti del .NET Framework, a partire da .NET 5.0.11 inclusa
☐	☒	si possono definire solo in progetti per la piattaforma .NET Core
☐	☒	devono avere property o metodi pubblici, altrimenti non servono a nulla
☐	☒	devono avere solo il costruttore di default altrimenti non sono staticamente corretti
☒	☐	una qualsiasi classe che estenda `System.Attribute` è un custom attribute
☒	☐	non sono estendibili, ovvero se una classe rappresenta un custom attribute non se ne può definire una specializzazione
☐	☒	una classe può avere (essere decorata da) al più un custom attribute
☒	☐	si può restringere il campo di applicazione di un custom attribute a solo alcuni elementi del linguaggio (classi, metodi...)

2. Supponete di avere un oggetto `o` di tipo `C` con accesso esclusivo a `xyz.mp4`

Vero	Falso	
☐	☒	appena `o` diventa irraggiungibile il garbage collector provvede a rilasciare il lock su `xyz.mp4`
☐	☒	appena `o` diventa irraggiungibile il garbage collector provvede a rilasciare lo spazio disco occupato da `xyz.mp4`
☐	☒	se `o` non viene usato in un costrutto using si ha errore statico
☒	☐	solo il costrutto using garantisce che il lock su `xyz.mp4` sia rilasciato correttamente
☒	☐	per essere staticamente corretta `C` deve implementare `IDisposable`
☒	☐	solo se `C` implementa `IDisposable` si può usare il costrutto using per oggetti di tipo `C`

3. Nell'ambito dei Versioning System e più specificamente di Git

Vero	Falso	
☐	☐	ogni commit su Git ha un singolo parent
☐	☐	un commit su Git può avere un numero arbitrario di figli, in particolare nessuno o più di uno
☐	☐	Git è basato su un modello client-server
☐	☐	Fare push può generare conflitti sul remote su cui si fa il push
☐	☐	Git prevede un sistema di garbage collection che fa pulizia dei commit diventati irraggiungibili
☐	☐	la versione di un file di cui si è fatto staging è quella che verrà aggiunta al prossimo commit anche se tra staging e commit il file viene modificato

4. Volendo definire una variabile `myDbSet` che rappresenta in memoria il risultato di una query effettuata usando l'EF.

Vero	Falso	
☐	☒	Una doppia enumerazione di `myDbSet` genera errore dinamico
☒	☐	Accedendo due volte all'i-esimo elemento di `myDbSet` si possono ottenere valori diversi
☒	☐	Scegliere `List<C>`, dove `C` è il tipo degli elementi risultanti dalla query, come tipo di `myDbSet` può richiedere più RAM di quella disponibile
☐	☒	Il tipo di `myDbSet` deve implementare `IDbSet`, altrimenti si ha un errore statico
☒	☐	Il tipo più naturale per `myDbSet` è `IQueryable` o un suo tipo derivato
☒	☐	Accedere direttamente (usando un indice o il metodo `ElementAt()`) a più elementi di `myDbSet` causa multiple enumerazioni
☐	☒	Ogni modifica a elementi di `myDbSet` viene automaticamente e immediatamente riflessa sul corrispondente elemento nella base di dati