Appello TAP del 7/02/2020

Scrivere nome, cognome e matricola sul foglio protocollo, indicando anche se avete nel piano di studi TAP da 6 CFU (quello attuale) o da 8 CFU (quello "vecchio"). Avete a disposizione tre ore.

Esercizio 1

Scrivere l'extension-method `Zip` che, dato un array `s` di sequenze di elementi di tipo `T`, dove `T` è un parametro di tipo, produce una sequenza di array di elementi di tipo `T`.

L'elemento i-esimo del risultato è l'array contenente nella cella j-esima, l'elemento i-esimo della j-esima sequenza in input.

Ad esempio, sull'array

$$[\ [1, 2, 3, 4],\ [10, 20, 30, 40],\ [100, 200, 300, 400]\ ]$$

si avrà come risultato la sequenza

$$[\ [1, 10, 100],\ [2, 20, 200],\ [3, 30, 300],\ [4, 40, 400]\ ]$$

Il metodo dovrà sollevare

- `ArgumentNullException` se `s` o uno dei suoi elementi è `null`;
- `ArgumentException` se gli elementi di `s` hanno lunghezze differenti.

Si noti che le sequenze usate come elementi del'array argomento di `Zip` (e quindi il suo risultato) possono essere infinite.

Esercizio 2

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

1. Input della chiamata sotto test: `s` deve essere un array (non nullo) di 3 elementi non nulli, di lunghezza diversa fra loro.

   Output atteso: eccezione di tipo `ArgumentException`

2. Esempio presentato nell'esercizio 1.

3. Test parametrico con parametro `approx` che indica il grado di precisione nella verifica del risultato.

   Input della chiamata sotto test: `s` deve essere un array di tre sequenze **infinite**

   - `[a, b, c, a, b, c, a, b, c,...]`
   - `[b, c, a, b, c, a, b, c, a, ...]`
   - `[c, a, b, c, a, b, c, a, b, ...]`

   dove `a`, `b` e `c` sono valori distinti di un tipo a vostra scelta.

   Il test deve verificare che i primi `approx` elementi del risultato coincidano con quelli attesi, ovvero siano `[a, b, c ]`, `[b, c, a ]`, `[c, a, b ]`, `[a, b, c ]`, `[b, c, a ]`, `[c, a, b ]`, ...

Parte a Quiz. Cognome e Nome:_____

Rispondere alle seguenti domande a risposta multipla. Fate attenzione che

- alcune [combinazioni di] risposte sono così sbagliate da portare *punteggio negativo*;
- alcune domande hanno più risposte corrette e per totalizzare il punteggio pieno dovete selezionarle *tutte*.

Segnare con **Y** le affermazioni vere, con **N** quelle false.

Esercizio 3

vincolo Where ==> IEnumerable

Data l'espressione LINQ

```
new []{26.85, 28.86, 29.1, 30.94, 31.02, 31.4, 35.61, 38.6, 39.79}.
    Where(i => i >= 30);
```

quali delle seguenti affermazioni sono vere?

- [ ] Il tipo dell'espressione è `IEnumerable<double>`
- [ ] Il tipo dell'espressione è `double[]`
- [ ] Il tipo dell'espressione è `IQueryable<double>`
- [ ] Il tipo dell'espressione è `double`
- [ ] L'espressione non è sintatticamente corretta perché il metodo `Where` si può invocare solo su un'espressione di tipo `DbSet<T>`
- [ ] L'espressione può essere assegnata ad una variabile di tipo `IQueryable<double>`
- [ ] Per poter assegnare l'espressione ad una variabile di tipo `IQueryable<double>` bisogna prima cambiarle il tipo, ad esempio usando il metodo `AsQueryable()`
- [ ] L'espressione può essere assegnata ad una variabile di tipo `double[]`
- [ ] All'espressione può essere assegnata un valore di tipo `double[]`
- [ ] Nessuna delle precedenti affermazioni è vera

Esercizio 4

Si consideri il seguente frammento di codice basato sull'entity framework.

```
public class Alpha {
    public int AlphaId { get; set; }
    public int BetaId { get; set; }
    public virtual ICollection<Beta> Betas { get; set; }
    public virtual ICollection<Alpha> Alphas { get; set; }
}
public class Beta {
    public int BetaId { get; set; }
    [MaxLength(50),Index(IsUnique = true)]
    public string Beta1 { get; set; }
    public virtual ICollection<Alpha> Alphas { get; set; }
}
public class MyContext : DbContext {
    public DbSet<Alpha> Alphas { get; set; }
    public DbSet<Beta> Betas { get; set; }
      /*...Constructors...*/
}
```

☑ L'entità `Alpha` è collegata all'entità `Beta` *esclusivamente* da una relazione molti a molti, rappresentata dalla coppia di proprietà `Alphas` (nella classe `Beta`) e `Betas` (nella classe `Alpha`)

☒ L'entità `Alpha` è collegata all'entità `Beta` *esclusivamente* da due relazione uno a molti, rappresentate rispettivamente dalle proprietà `Alphas` (nella classe `Beta`) e `Betas` (nella classe `Alpha`)

☒ La proprietà `BetaId` nella classe `Alpha` è una proprietà di navigazione verso l'entità `Beta`, cioè nella tabella generata sarà una chiave esterna verso la tabella che rappresenta l'entità `Beta`

☑ la proprietà `Alphas` nella classe `Alpha` rappresenta una auto-relazione sull'entità `Alpha`

☒ la proprietà `Alphas` nella classe `Alpha` è equivalente alla proprietà `Alphas` del contesto, nel senso che entrambe restituiscono sempre tutti gli oggetti di tipo `Alpha` noti

☒ la base di dati generata dall'entity framework per questo frammento contiene le due tabelle per `Alphas` e `Betas` e nessun'altra

☑ la base di dati generata dall'entity framework per questo frammento contiene le due tabelle per `Alphas` e `Betas` e una per la relazione molti a molti fra `Alpha` e `Beta` e nessun'altra

☒ la base di dati generata dall'entity framework per questo frammento contiene più di tre tabelle.

☑ nella base di dati generata dall'entity framework per questo frammento, la tabella per `Alphas` ha un'unica chiave esterna verso se stessa.

☒ Il seguente frammento di codice può sollevare un'eccezione dovuta a violazione di chiave (secondaria)?

```
using (var c = new EntityUnderstanding.MyContext(ConnectionString)) {
    if (c.Betas.Any(b=>b.Beta1=="Paperino"))
        throw new ApplicationException("name already in use");
    var beta = c.Betas.Create();
    beta.Beta1 = "Paperino";
    c.Betas.Add(beta);
    c.SaveChanges();
}
```