Appello TAP del30/1/2023

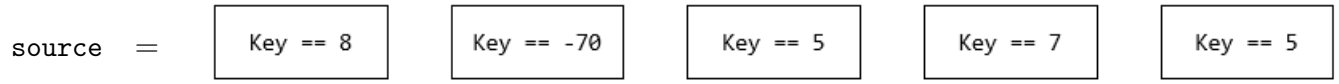
Scrivere nome, cognome e matricola sul foglio protocollo. Avete a disposizione due ore e mezza.

Esercizio 1 (8 punti)

Si consideri l'interfaccia

```
public interface IIdentified { public int Key { get; } }
```

Scrivere l'extension-method `Lookup` che, invocato su `db`, una sequenza (eventualmente nulla) di elementi di un tipo generico `T` che estende `IIdentified`, e su `what`, un intero, restituisce la prima posizione in cui si trova un elemento con `what` come `Key`. Nel caso in cui `db` non contenga nessun elemento con `what` come `Key`, se `db` è finita restituirà null. Altrimenti il suo comportamento non è definito (=potete fare quello che volete). Per esempio, sulla sequenza



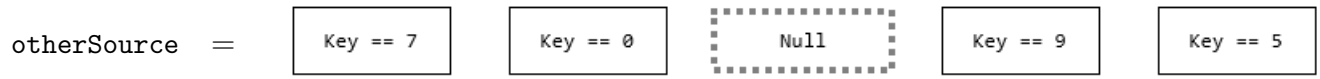
`source.Lookup` (8) == 0, `source.Lookup` (5) == 2, `source.Lookup` (11) == null

Il metodo dovrà prendere come parametro "this" `db`, la sequenza sorgente, **che può anche essere infinita**, e un intero `what`.

Il metodo deve sollevare l'eccezione `ArgumentNullException` se si verifica una delle seguenti condizioni

- `db` è null
- uno degli elementi in `db` prima di quello con `what` come `Key` è nullo.
- uno degli elementi in `db` è nullo e `db` non contiene nessun elemento con `what` come `Key`

Per esempio, sulla sequenza



`otherSource.Lookup` (0) == 1, ma `otherSource.Lookup` (9) e `otherSource.Lookup` (11) devono sollevare `ArgumentNullException`.

La dichiarazione del metodo deve tenere conto degli aspetti di nullabilità dei tipi reference nella scelta dei tipi e usare opportuni vincoli per limitarne l'istanziabilità ai soli tipi che estendono/implementano `IIdentified`.

Esercizio 2 (8 punti)

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

1. Implementare il primo esempio (quello relativo alla chiamata su 8 e la sequenza `source`)
2. Input della chiamata sotto test: `db` è una sequenza infinita di elementi tutti aventi `Key` diverso da 42 e contenente un elemento nullo in posizione 42, `what` è 42.

   Output atteso: la chiamata deve sollevare `ArgumentNullException`.
3. Test parametrico con un parametro intero `size`. Se `size` non è maggiore o uguale a 20 il test dovrà risultare *inconclusive*.

   Input della chiamata sotto test: `db` è una sequenza di lunghezza `size` di elementi aventi `Key` progressive a partire da 0 (cioè 0, 1, 2, ... 99) e `what` == 10.

   Il test dovrà verificare che la property `Key` sia stata invocata almeno una volta sui primi 11 elementi di `db` (cioè quelli aventi `Key` da 0 a 10) e nessuna volta sui successivi.

Esercizio 3 (9 punti)

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

1. Si considerino le seguenti classi dove `AnException` è una classe che estende `Exception`.

```
public class C {
    public int P { get; init; }
    public C(int x = 42) { P = x; }
    public IEnumerable<bool> M(int a) {
        for (int i = 0; i < a; i++) yield return true;
        if (a==P) throw new AnException();
        while (true) yield return false;
    }
}
[TestFixture]
public class CTest {
    [Test]
    public void T1() { Assert.That(new C().M(7), Is.Not.Empty); }
    [Test]
    public void T2() { Assert.That(new C(7).M(7), Is.Not.Empty); }
    [Test]
    public void T3() { Assert.That(
        new C(7).M(7), Throws.TypeOf<AnException>());
     }
    [Test]
    public void T4() {  Assert.That(
       () => new C(7).M(7), Throws.TypeOf<AnException>());
    }
    [Test]
    public void T5() {  Assert.That(
       () => new C(7).M(7).Any(), Throws.TypeOf<AnException>());
    }
    [Test]
    public void T6() {  Assert.That(
        () => new C(7).M(7).Take(10), Throws.TypeOf<AnException>());
    }
    [Test]
    public void T7() {  Assert.That(
        () => new C(7).M(7).ToArray(), Throws.TypeOf<AnException>());
    }
    [Test]
    public void T8() {  Assert.That(
        new C(7).M(7).ToArray(), Throws.TypeOf<AnException>());
    }
    [Test]
    public void T9() {
        var source = new C() { P = 57 };
        Assert.That(source.M(57).Take(7).ToArray(),
            Is.EqualTo(new[]{true, true, true, true, true, true, true}));
    }
}
```

Vero	Falso	
☐	☐	Il test `T1` non compila perché manca il parametro nel costruttore
☐	☐	Il test `T1` ha successo
☐	☐	Il test `T1` fallisce perché la chiamata solleva eccezione
☐	☐	Il test `T2` ha successo
☐	☐	Il test `T3` ha successo
☐	☐	Il test `T4` ha successo
☐	☐	Il test `T5` ha successo

☐ ☐ Il test `T6` ha successo
☐ ☐ Il test `T7` ha successo
☐ ☐ Il test `T8` ha successo
☐ ☐ Il test `T9` ha successo
☐ ☐ Il test `T9` non compila perché la property `P` é di sola lettura

2. Si consideri il seguente frammento di un progetto che usa l'Entity Framework

```
[Index(nameof(Login), IsUnique = true)]
public class Entity1 {
    public int Id { get; set; }
    public string Login { get; set; }
    public int MyKey { get; set; }
    public List<Entity3> Entities3 { get; set; }
}
public class Entity2 {
    public int Entity2Id { get; set; }
    public Entity1? Entity1 { get; set; }
}
public class Entity3 {
    [Key]
    public int MyKey { get; set; }
    public int Entity2Id { get; set; }
    public int Entity1Id { get; set; }
    public List<Entity1> Entities1 { get; set; }
}
public class Entity4 {
    public int Entity4Id { get; set; }
    public Entity2 Entity2 { get; set; }
}
public class MyDbContext : DbContext {
    public DbSet<Entity1> Entity1s { get; set; }
    public DbSet<Entity3> Entity3s { get; set; }
}
```

Vero Falso

☐ ☐ l'EF non può costruire il DB per questo frammento perché manca l'overriding di `OnModelCreating`
☐ ☐ la classe `Entity1` è un'entità, ovvero ha una tabella corrispondente sul DB
☐ ☐ la classe `Entity2` è un'entità, ovvero ha una tabella corrispondente sul DB
☐ ☐ la classe `Entity3` non è un'entità, perché il nome della chiave non segue le convenzioni
☐ ☐ la classe `Entity4` è un'entità, ovvero ha una tabella corrispondente sul DB
☐ ☐ per rappresentare questo frammento, l'EF genera meno di 4 tabelle sul DB
☐ ☐ per rappresentare questo frammento, l'EF genera esattamente 4 tabelle sul DB
☐ ☐ per rappresentare questo frammento, l'EF genera più di 4 tabelle sul DB
☐ ☐ l'EF genera almeno una tabella che non rappresenta nessuna delle entità nel frammento
☐ ☐ il seguente codice solleva eccezione per violazione di indice *unique*:
`var x = new Entity1() { Login = "puffo"}; var y = new Entity1() { Login = "puffo"};`
☐ ☐ nella tabella associata a `Entity1` ci possono essere più righe con lo stesso valore di `Login`
☐ ☐ le property `Entities1` in `Entity3` e `Entities3` in `Entity1` rappresentano i due lati di una relazione molti-a-molti per l'EF
☐ ☐ le property `Entities1` in `Entity3` e `MyKey` in `Entity1` rappresentano una relazione uno-a-molti per l'EF
☐ ☐ la property `Entity1Id` nella classe `Entity3` è una chiave esterna su `Entity1`