Appello TAP del 02/02/2018

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 due ore.

Esercizio 1 (10 punti)

Date le interfacce

```
public interface IExam{
    string Name { get; set; }
    DateTime Date { get; set; }
    int Grade { get; set; }
    IStudent Student { get; set; }
}

public interface IStudent {/*....*/ }
```

- Scrivere l'extension-method `Career` che filtra da una sequenza di esami quelli sostenuti da uno specifico studente.

  Il metodo dovrà prendere come parametri:

  - "this" `register`, la sequenza sorgente (il registro degli esami)
  - `s`, lo studente di cui si vogliono gli esami.

  Il metodo `Career` deve sollevare l'eccezione `ArgumentNullException` se `register` o `s` sono `null`.

- Scrivere l'extension-method `LastRegistration` che seleziona da una sequenza di esami quello sostenuto più recentemente da uno specifico studente e restituisce `null` se non risultano esami sostenuti dallo studente. Il metodo dovrà prendere come parametri:

  - "this" `register`, la sequenza sorgente (il registro degli esami)
  - `s`, lo studente di cui si vogliono gli esami.

  Il metodo `LastRegistration` deve sollevare l'eccezione `ArgumentNullException` se `register` o `s` sono `null` e demandare, per quanto possibile, le computazioni al provider dei dati. Se lo ritenete opportuno potete definire versioni di `LastRegistration` in overloading.

- È possibile dare implementazioni dei metodi richiesti ai punti precedenti in grado di terminare anche in caso le sequenze sorgente siano infinite?

Esercizio 2 (10 punti)

Modificare il seguente codice in modo da renderne facile lo unit testing

```
public class B {
    public int Y { get; }
    public int X { get; }
    public B() {
        X = 3;
        Y = 7;
    }
    public B(int x) {
        X = x;
        Y = 2 * x;
    }
}

public class A {
    public B MyB { get; set; }
    public A() {MyB = new B();}
    bool M1(int a, int b) {
        if (a > MyB.X) return false;
        if (b < MyB.Y) return false;
        return true;
    }
    public B M2(int x) { return new B(x);}
}
```

Esercizio 3 (10 punti)

Si consideri il seguente sistema:

```
public interface ID {
    //return the expected value for a D
    int CurrentValue();
}

public interface IC {
    // An oracle to guess D values
    ID Oracle { get; }
    // returns....IRRILEVANTE AI FINI DI QUESTO ESERCIZIO
    // throws ArgumentException if x less than zero
    int M(int x);
    // returns true if x is greater than Oracle.CurrentValue()
    bool MeetsExpectations(int x);
}

public class C : IC {
    public C(ID d) {
        Oracle = d;
    }
    public ID Oracle { get; }
    public int M(int x) {/*....*/}
    public bool MeetsExpectations(int x) {/*....*/}
}

public class D : ID {
    public int CurrentValue() {/*....*/}
}
```

Implementare, senza usare Moq o altri testing framework, i seguenti test su `C` per verificare (rispettivamente) che

1. la chiamata di `M` su -7 sollevi una `ArgumentException`;
2. la chiamata di `MeetsExpectations` su 20 restituisca `true` se `Oracle.CurrentValue()` restituisce 10;
3. una chiamata di `MeetsExpectations` richieda una e una sola chiamata a `Oracle.CurrentValue`.