Appello TAP del 7/09/2018

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)

Scrivere l'extension-method `apply` generico su un parametro di tipo `T` che date due sequenze di elementi di tipo `T` e una funzione che accetta due parametri di tipo `T` e restituisce un elemento di tipo `T` produce una sequenza di elementi di tipo `T`, applicando la funzione presa come argomento agli elementi delle due sequenze.

Il metodo dovrà prendere come parametri

- "this" `first`, la sequenza di elementi di tipo `T` da usare come primo argomento delle chiamate a `f`
- `second`, la sequenza di elementi di tipo `T` da usare come secondo argomento delle chiamate a `f`
- `f` la funzione usata per calcolare i valori della sequenza risultato.

Ad esempio sulle sequenze `first` $=1,-6,7$ e `second` $=0,-5,7$ e la funzione somma produrrà la sequenza $1,-11,14$, mentre sulle sequenze `first` =`"Hello "`, `"Ciao "`, `"Bonjour "`, `"Hallo "` e `second` = `"World"`, `"Mondo"`, `"le Monde"`, `"Welt"` e la funzione concatenazione produrrà la sequenza `"Hello World"`, `"Ciao Mondo"`, `"Bonjour le Monde"`, `"Hallo Welt"`.

In caso uno dei parametri che rappresentano una sequenza di elementi di tipo `T` abbia un numero di elementi strettamente minore dell'altra, gli argomenti mancanti per la chiamata di funzione verranno sostituiti con il valore di default del tipo `T`.

Quindi, ad esempio sulle sequenze `first` $=1,2$ e `second` $=10,20,30$ e la funzione somma produrrà la sequenza $11,22,30$ (perché il valore di default degli interi è 0), mentre sulle sequenze `first` =`"Hello "`, `"Ciao"`, `"Bonjour"` e `second` = `"World"` e la funzione concatenazione produrrà la sequenza `"Hello World"`, `"Ciao"`, `"Bonjour"`, `"Hallo"` (perché il valore di default delle stringhe è `null` e concatenare `null` ad una stringa restituisce la stringa stessa).

Il metodo `apply` deve sollevare `ArgumentNullException` se almeno uno dei suoi parametri è `null`.

Si noti che le sequenze rappresentate da entrambi gli argomenti di `apply` (e quindi il suo risultato) possono essere infinite.

Esercizio 2 ([2+3+5] = 10 punti)

Implementare, usando NUnit e/o Moq, i seguenti test relativi al metodo `apply`, dell'esercizio 1.

1. Input della chiamata sotto test: `first` deve essere la sequenza $1,2,3$, `second` deve essere la sequenza $10,20,30$ e la funzione deve essere nulla.

   Output atteso: deve essere sollevata l'eccezione `ArgumentNullException`.

2. Input della chiamata sotto test: `first` deve essere la sequenza `"Hello "`, `"Ciao "`, `"Bonjour "`, `"Hallo "`, `second` deve essere la sequenza `"World"`, `"Mondo"`, `"le Monde"`, `"Welt"` e la funzione deve essere la concatenazione di stringhe.

   Output atteso: la sequenza `"Hello World"`, `"Ciao Mondo"`, `"Bonjour le Monde"`, `"Hallo Welt"`.

3. Test parametrico con parametro `howMany` che indica quanti valori del risultato devono essere verificati.

   Input della chiamata sotto test: `first` deve essere una sequenza non nulla e non vuota e `second` una sequenza **infinita** a vostra scelta, purché generino come risultato di `apply` una sequenza di elementi tutti diversi fra loro.

   Il test deve verificare che i primi `howMany` elementi del risultato coincidano con quelli attesi (che dipendono dalla scelta di argomenti che avete fatto).

Esercizio 3 (10 punti)

Implementare la classe `ComplexNumber` che rappresenta i numeri complessi con le quattro operazioni (somma, sottrazione, moltiplicazione e divisione), l'uguaglianza e conversione implicita da reale a complesso ed esplicita da complesso a reale (corretta solo se la parte immaginaria è zero), in modo tale che il seguente frammento di codice sia staticamente corretto

```
var c1 = new ComplexNumber(3,10);
double x = c1.Re;
double y = c1.Im;
ComplexNumber c2 = c1 + c1;
double err = (double) c1;
ComplexNumber c3 = 6.8;
c1 = (c2 - c1) / c3;
c2 = c1 * c3;
```

Siccome i `double` sono approssimati, per verificare uguaglianze fra valori si definisca e usi una costante di approssimazione, ad esempio `approx` ==0.000001, e si considerino uguali valori che differiscono fra loro, in valore assoluto, meno di `approx`.

Per chi si fosse dimenticato come funzionano i numeri immaginari:

- $(a + bi) + (c + di) == (a + c) + (b + d)i$
- $(a + bi) - (c + di) == (a - c) + (b - d)i$
- $(a + bi) * (c + di) == (ac - bd) + (ad + bc)i$
- $(a+bi)/(c+di) == \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$, se $c! = 0$ e/o $d! = 0$ (se entrambi sono 0 si ha errore divisione per 0).
- $(a + bi) == (c + di)$ se e solo se $a == c$ e $b == d$.