

## Appello TAP del 9/9/2021

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

### Esercizio 1 (9 punti)

Sia  $T$  un tipo reference *ordinato*, cioè che implementa l'interfaccia `Comparable` e che quindi ha il metodo `CompareTo(Object)` che restituisce un numero intero minore di 0 se l'istanza corrente precede l'argomento della chiamata, 0 se sono uguali e un numero maggiore di 0 se lo segue (rispetto all'ordine).

Scrivere l'extension-method `MinUpToNow` che, invocato su `leftSeq`, una sequenza di elementi di tipo  $T$  e un ulteriore parametro, `rightSeq`, anch'esso una sequenza di elementi di tipo  $T$ , restituisce la sequenza di elementi di tipo  $T$  il cui elemento in posizione  $i$  è il minimo fra gli elementi in posizione da 0 a  $i$  delle due sequenze.

Per esempio, il seguente frammento di codice

```
foreach (var d in
    new [] { "qui", "quo", "qua" }.MinUpToNow(new [] { "tip", "pippo", "pluto" }))
    Console.Write("{0}, ", d);
Console.WriteLine("finito");
```

stampa:

```
"qui", "pippo", "pippo", finito
```

Il metodo dovrà prendere come parametro “this” `leftSeq`, la sequenza sorgente, e come altro parametro la sequenza `rightSeq`. Nota: entrambe le sequenze possono anche essere infinite.

Il metodo deve sollevare l'eccezione...

- `ArgumentNullException` se `rightSeq` o `leftSeq` o uno dei loro elementi sono `null`;
- `ArgumentException` se `rightSeq` contiene meno elementi di `leftSeq` o viceversa `leftSeq` contiene meno elementi di `rightSeq`.

La dichiarazione del metodo deve prevedere opportuni constraint per garantire che il tipo generico soddisfi le condizioni indicate.

### Esercizio 2 ([1+2+4] = 7 punti)

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

1. Input della chiamata sotto test: `leftSeq` è la sequenza "bianco", "rosso", "verde", `rightSeq` è una qualsiasi sequenza **infinita** di stringhe  
Output atteso: una `ArgumentException`.
2. Input della chiamata sotto test: `leftSeq` è la sequenza "qui", "quo", "qua", "paperino", "paperone" e `rightSeq` è la sequenza "topolino", "pippo", "pluto", "tip", "tap"  
Output atteso: la sequenza "qui", "pippo", "pippo", "paperino", "paperino"
3. Test parametrico con un parametro di tipo intero, `errorIndex`.  
Input della chiamata sotto test: `leftSeq` è la sequenza **infinita** i cui elementi sono tutti la stringa "rosa" e `rightSeq` è la sequenza **infinita** i cui elementi sono tutti la stringa "viola", tranne quello in posizione `errorIndex` (contando da 0) che deve essere `null`.  
Output atteso: una `ArgumentNullException`.

### Esercizio 3 (9 punti)

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

1. In C#, nelle intestazioni dei metodi, le eccezioni sollevate

Vero Falso

- ☐ ☐ Non si devono dichiarare
- ☐ ☐ Si devono dichiarare *tutte* con throws
- ☐ ☐ Si devono dichiarare con throws *solo* quelle *user defined*

2. Se un oggetto di classe C ha bisogno di un logger di tipo L, secondo la dependency-injection:

Vero Falso

- ☐ ☐ C deve usare la reflection per ottenere un'istanza di L
- ☐ ☐ L deve fornire un costruttore senza parametri, in modo che C possa fare `new L()`
- ☐ ☐ I costruttori di C devono avere un parametro di tipo L

3. L'esecuzione del comando `using (T x=e) { ... }` corrisponde grosso modo a quella di:

Vero Falso

- ☐ ☐ `T x=e; try { ... } finally { x.Dispose(); Delete(x); }`
- ☐ ☐ `try {...} finally { x.Dispose(); Delete(x); }`
- ☐ ☐ `T x=e; try { ... } finally { x.Dispose(); }`

4. Se x è un `IQueryable`, le seguenti espressioni si possono usare come `IEnumerable`

Vero Falso

- ☐ ☐ `x`
- ☐ ☐ `x.AsEnumerable()`
- ☐ ☐ `new IEnumerable(x)`

5. In Git i seguenti comandi richiedono la connessione al server:

Vero Falso

- ☐ ☐ `git log`
- ☐ ☐ `git clone`
- ☐ ☐ `git commit`

6. Nel testing, si usano

Vero Falso

- ☐ ☐ Gli stub per lo state-based testing, i mock per l'interaction-based testing
- ☐ ☐ I mock per lo state-based testing, gli stub per l'interaction-based testing
- ☐ ☐ Indifferentemente, stub e mock (che sono fra loro sinonimi)

7. Per definire un custom-attribute si deve

Vero Falso

- ☐ ☐ Scrivere un file XML
- ☐ ☐ Usare l'ADO Entity Framework
- ☐ ☐ Scrivere una classe

8. Per il passaggio di parametri per riferimento in C#

Vero Falso

- ☐ ☐ Si usa la keyword `ref` sia nella dichiarazione del parametro, sia nella chiamata
- ☐ ☐ Si usa la keyword `ref` solo nella dichiarazione del parametro
- ☐ ☐ Si usa la keyword `ref` solo quando si passa l'argomento al momento della chiamata

9. In uno unit-test, ci aspettiamo che:

Vero Falso

- ☐ ☐ Ci sia un'unica asserzione, alla fine del metodo
- ☐ ☐ Ci siano tante asserzioni, una per ogni proprietà verificata dal test
- ☐ ☐ Ci sia un'asserzione dopo ogni istruzione in modo da tracciare dove fallisce