# Exercises: Unit Testing Methods, Arrays and Lists

Submit your solutions here: <https://judge.softuni.org/Contests/4682/Unit-Testing-Methods-Arrays-and-Lists-Exercise>

# Unit Test Method: Fibonacci

Look at the **provided skeleton** and examine the Fibonacci.cs class that you will test:A screenshot of a computer

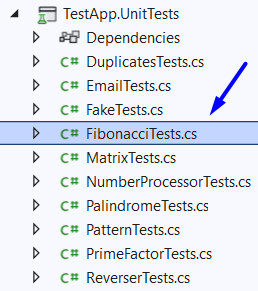
Description automatically generated

The method takes in an **integer** , and calculates the **Fibonacci** **number**:

A screenshot of a computer code

Description automatically generated

Then, look at the tests inside the FibonacciTests.cs class:



A screenshot of a computer program

Description automatically generated

The first test if **finished** so you have a **reference**, the rest of the tests are **empty,** and your task is to finish them. The tests should run when you're finished:

A screenshot of a computer

Description automatically generated

# Unit Test Method: Email

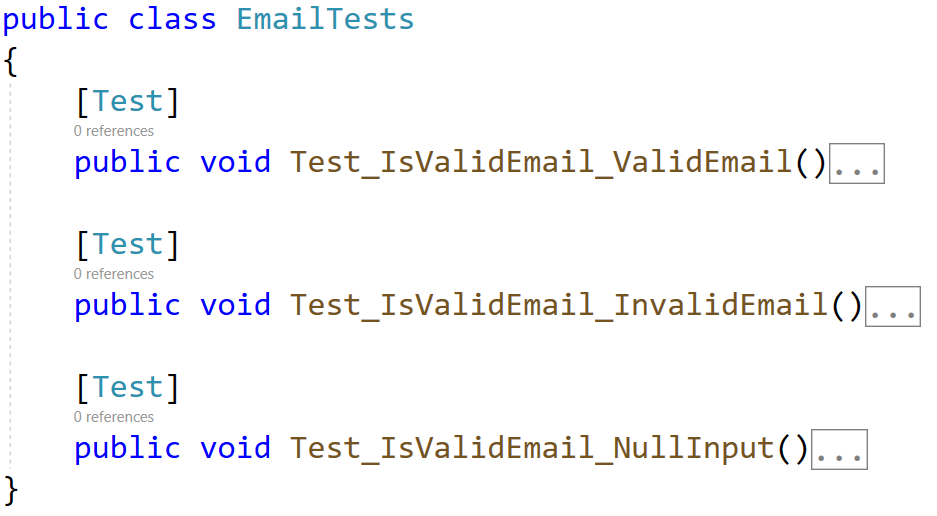
Test a given method which takes in a **string representing an email address** and finds if the given email is indeed **a valid email or not**.

The method is found in the Email.cs file:

A screen shot of a computer code

Description automatically generated

You are given again a **test** **file** EmailTests.cs which contains **3 tests**. One of them has been **finished partially**, and **two** are **empty** for you to finish:



When you are ready make sure your **tests run:**

A screenshot of a computer

Description automatically generated

# Unit Test Method: Prime Factor

Test a given method which takes in a **long number** and finds the largest prime factor of the given number.

The method is found in the PrimeFactor.cs file:

A screenshot of a computer program

Description automatically generated

A close up of a text

Description automatically generated

In the **test** **file** PrimeFactorTests.cs you are given **3 empty tests**. Finish them and run them:

A screenshot of a computer

Description automatically generated

# Unit Test Array: Reverser

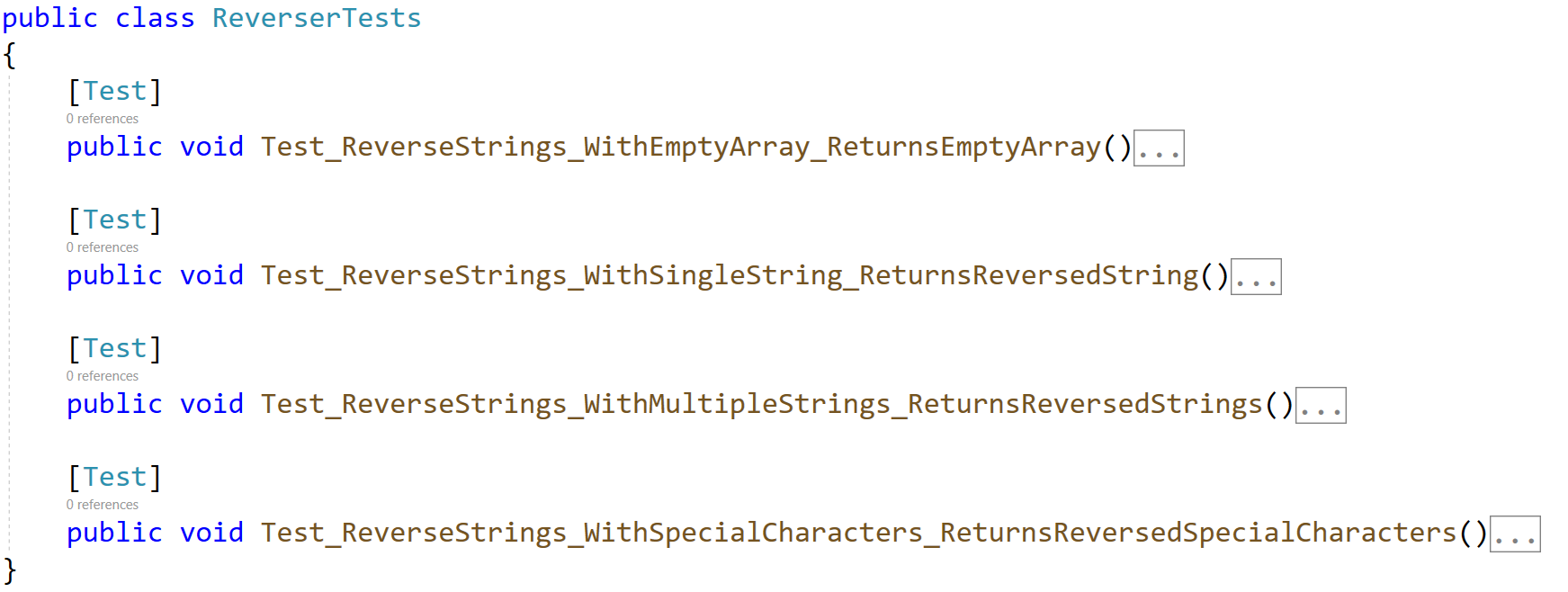
Test a given method which takes in an **array of strings** and **reverses** each string in thearray.

The method is found in the Reverser.cs file:

A close-up of a computer code

Description automatically generated

In the **test** **file** ReverserTests.cs you are given **1 finished test**, **1 partially finished test**, and **2 empty tests**. Finish them and run them:



A screenshot of a computer program

Description automatically generated

# Unit Test Array: Duplicates

Test a given method which takes in an **array of integers** and **removes all duplicate numbers**.

The method is found in the Duplicates.cs file:

A screenshot of a computer code

Description automatically generated

In the **test** **file** DuplicatesTests.cs you are given **2 partially finished test**, and **2 empty tests**. Finish them and run them:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Unit Test Array: Fake

Test a given method which takes in an **array of characters** and **removes all characters which are numbers and not letters.**

The method is found in the Fake.cs file:

A screenshot of a computer program

Description automatically generated

Notice the method throws ArgumentException if the array is **null**.

In the **test** **file** FakeTests.cs you are given **3 empty tests**:

A screenshot of a computer program

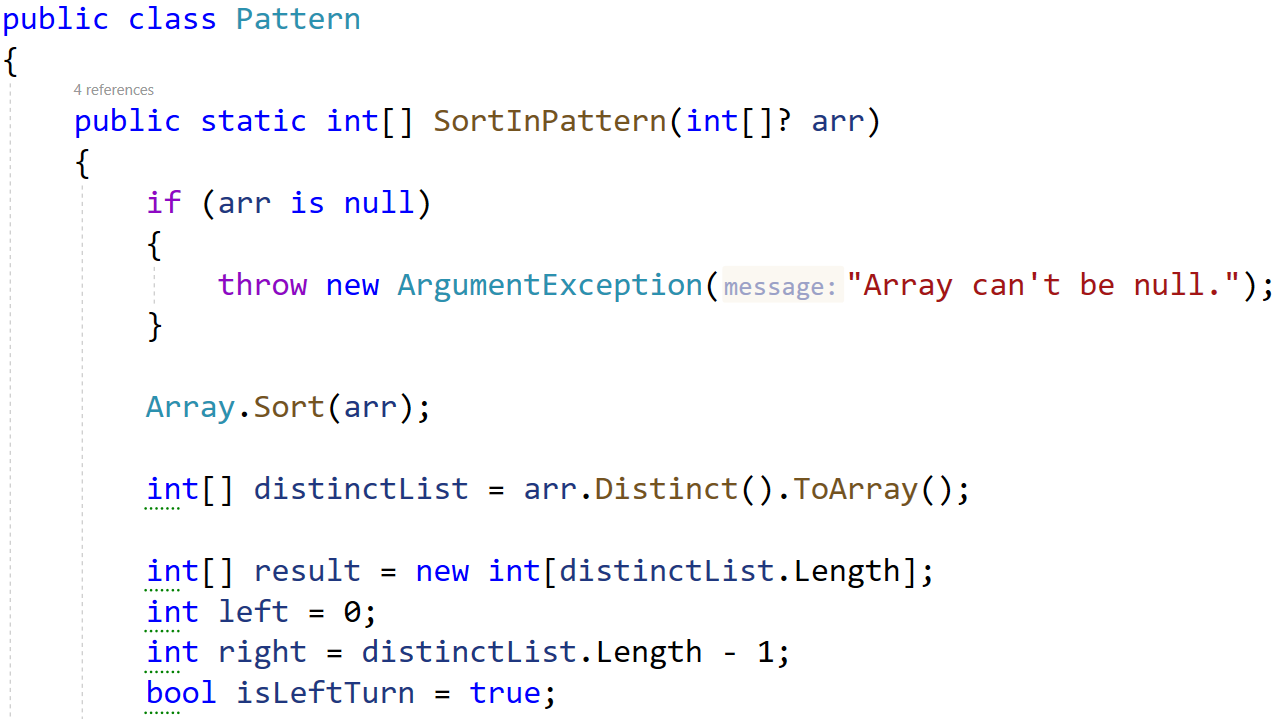
Description automatically generated

# Unit Test Array: Pattern

Test a given method which takes in an **array of integers** that **removes all duplicates** and **sorts** the list in a zig-zag pattern.

Example: If we have an array 1 2 1 3 4 10 12 15 the sorted array would be: 1 15 2 12 3 10 4.

The method is found in the Pattern.cs file:



A screenshot of a computer code

Description automatically generated

In the **test** **file** PatternTests.cs you are given **tests**:

A screenshot of a computer program

Description automatically generated

# Unit Test List: Number Processor

Test a given method which takes in a **list of integers** and adds each number in a new list following these principles:

* If the number is **even**, it **squares** it.
* If the number is **odd**, it **square** **roots** it**.**

The method is found in the NumberProcessor.cs file:

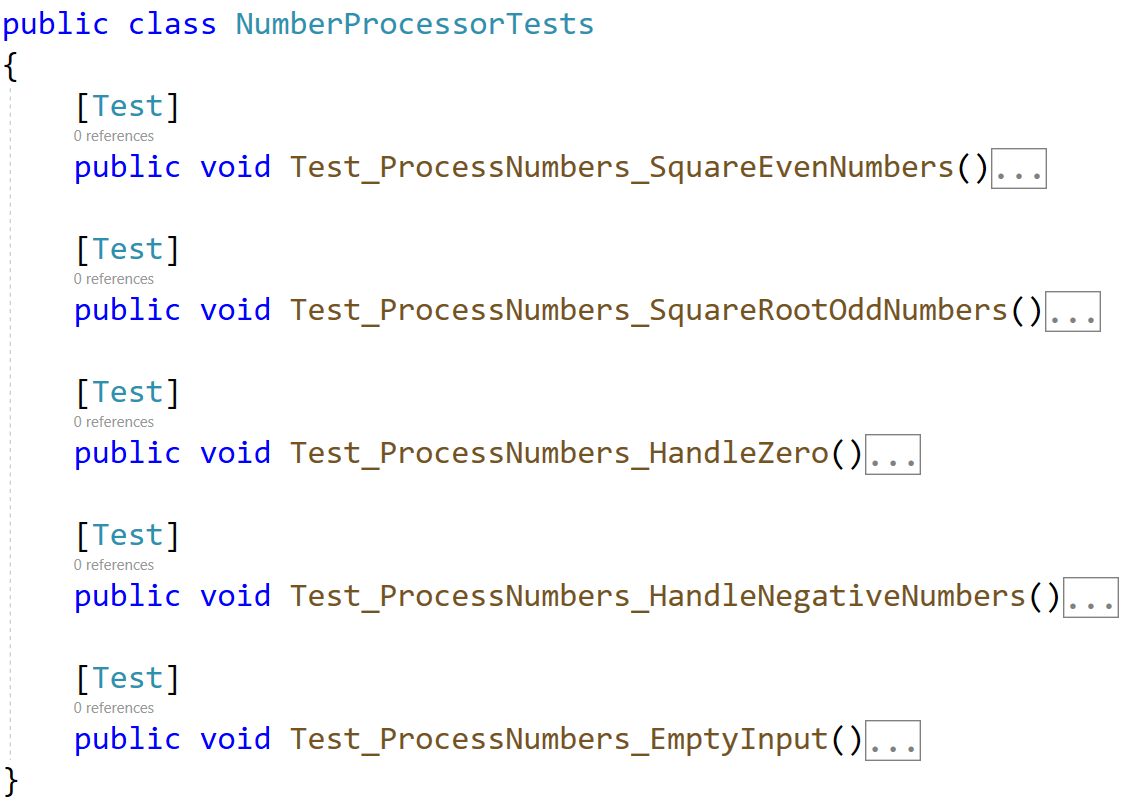
A screenshot of a computer code

Description automatically generated

A computer screen shot of a code

Description automatically generated

In the **test** **file** NumberProcessorTest.cs you are given **tests**. Finish them and run them:



# Unit Test List: Palindrome

Test a given method which takes in a **list of strings** and checks if every word in the array is a **palindrome**.

The method is found in the Palindrome.cs file:

A computer screen shot of a program

Description automatically generated

In the **test** **file** PalindromeTests.cs you are given **2 partially finished test**, and **3 empty tests**. Finish them and run them:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

# \* Unit Test List: Matrix

Test a given method which takes in **2** **lists of list of integers (matrix)** that performs **matrix addition** on them.

The method is found in the Matrix.cs file:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

Notice the method throws ArgumentException if the matrices are **not the** **same length**.

In the **test** **file** MatrixTests.cs you are given **2 partially finished test**, and **3 empty tests**:

A screenshot of a computer

Description automatically generated

Check the tests run successfully:

A screenshot of a computer

Description automatically generated