# Exercise: Unit Testing Methods

Problems for exercise and homework for the ["Programming Fundamentals and Unit Testing" course @ SoftUni](https://softuni.bg/trainings/4256/programming-fundamentals-and-unit-testing-september-2023)  
You can check your solutions in [Judge](https://judge.softuni.org/Contests/4357/Unit-Testing-Simple-Methods-Exercise)

## NUnit - Smallest of Three Numbers

In this task, you will **practice unit testing** by **creating test cases** for the **SmallestNumber** method in the **NumberProcessor** class. You have the already implemented method that takes three integer numbers as input and **returns the smallest of them**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Vowels Count

In this task, you will **practice unit testing** by **creating test cases** for the **PrintVowels** method in the **StringProcessor** class. You have the already implemented method that takes that receives a single string and **counts the number of vowels in it**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Characters in Range

In this task, you will **practice unit testing** by **creating test cases** for the **PrintCharacters** method in the **ASCIIPrinter** class. You have the already implemented method that receives two characters and **generates all the characters between them according to ASCII** (on a single line). If the **second letter's ASCII value is less than that of the first one**, then the two initial **letters should be swapped**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Password Validator

In this task, you will **practice unit testing** by **creating test cases** for the **ValidatePassword** method in the **Validator** class. You have the already implemented method checks if a given **password is valid** based on certain rules. The rules include the **password length**, **character types**, and the **presence of at least 2 digits**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Middle Characters

In this task, you will **practice unit testing** by **creating test cases** for the **ReturnMiddle** method in the **CharactersProcessor** class. You have the already implemented method that receives a single string and returns **the middle character(s) of the string**. If the string **length is even, there will be two middle characters**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUNit - NxN Matrix

In this task, you will **practice unit testing** by **creating test cases** for the **PrintMatrix** method in the **Printer** class. You have the already implemented method that receives a single integer **N** and generates an **NxN matrix** using this **number as a filler**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Factorial Division

In this task, you will **practice unit testing** by **creating test cases** for the **GetFactorial** and **FactorialDivision** methods in the **FactorialProcessor** class. You have the already implemented method that **calculates the factorial of two numbers**, **divides the first result by the second**, and returns the result formatted to the second decimal point. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Palindrome Integers

In this task, you will **practice unit testing** by **creating test cases** for the **Check** method in the **PalindromeDefiner** class. You have the already implemented method that checks **if a positive integer is a palindrome** (reads the same forwards and backwards). Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.

## NUnit - Top Number

In this task, you will **practice unit testing** by **creating test cases** for the the **LeastOneOdd** and **MultipleBy** methods in the **NumberAnalyser** class. You have the already implemented method that finds and returns **all top numbers in the range [1…n]**. A top number is defined based on **specific properties related to its digits**. Now, your goal is to **ensure that this method works correctly by writing unit tests using NUnit**.