IDENTIFICATION: Regular Expression #2 / PELE

<u>Overall Purpose</u>
The overall purpose for the group of 'Regular Expressions' assignments is to be able to use Regular Expressions for validation, finding, extracting and replacing text.

The whole group of assignments consist of two steps:

1. A basic understanding of Regular Expressions.
2. Using Regular expressions for validation, finding, extracting and replacing text. (this assignment)

<u>Background Material:</u>
- Overall view Wikipeia: https://en.wikipedia.org/wiki/Regular_expression
- Nice presentation (though not C#) https://cs.lmu.edu/~ray/notes/regex/
- Microsoft C# - quick reference: https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference
- C# How to use: https://www.c-sharpcorner.com/article/c-sharp-regex-examples/
- Microsoft reference Regex-class: https://docs.microsoft.com/en-us/dotnet/api/system.text.regularexpressions.regex?view=netcore-3.1

This Assignment: Regular Expression #2

Purpose
The purpose of this assignment is to obtain skills in validation, finding, extracting and replacing text using regular expressions.

Mission
Making experiments with
1. Validation (formula field validation).
2. UnitTest of validation class.
3. Finding, extracting and replacing text.

Assignment 1: Create validation using regular expressions

Create a .Net Core library, and create a class (remember to make it public) name it e.g. MyValidator.

You should make different methods to validate input from a virtual formula like the illustration below:
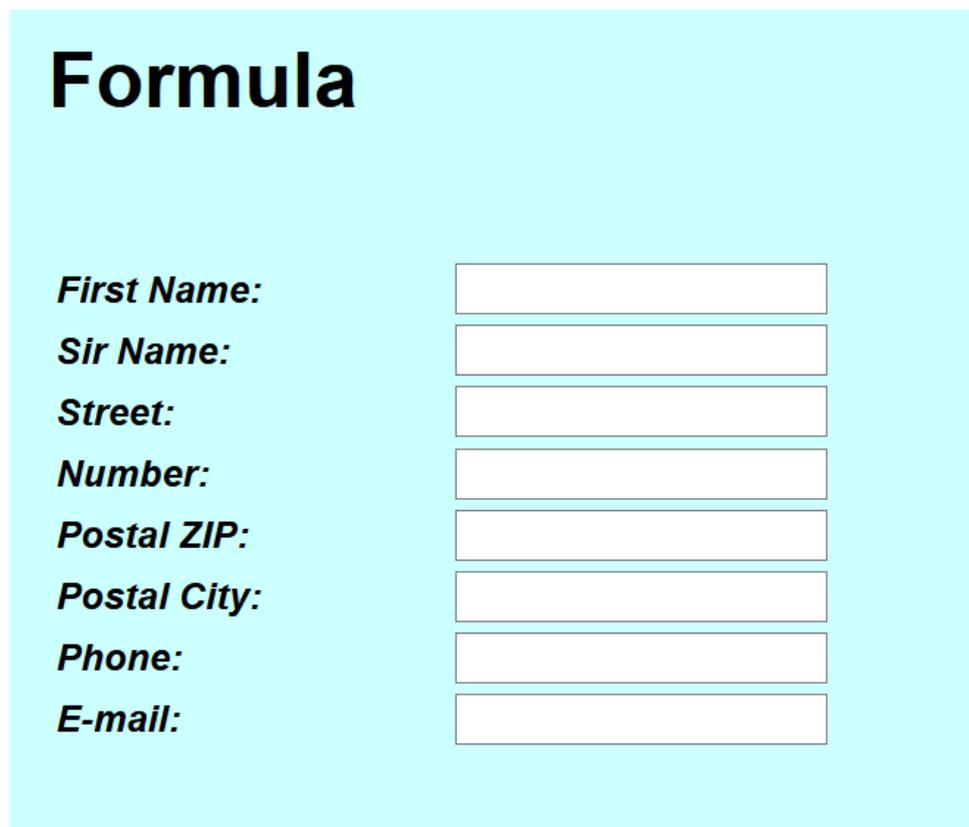
Constraints

- First Name, Sir Name, Street and Postal City are letters without any digits and at least one character long.

- Number could be the values 1-999 but no longer than three digits although it could be followed by several letters incl. digits e.g. 167 B, 1 i.e. number 167 B 1st flour
- Zip is always 4 digits
- Phone is always 8 digits long, but could be prefixed with e.g. +45 and perhaps a space
- E-mail is the normal letters, digits, hyphen, dot, underscore then '@' and the domain

Make five validator methods to fulfil these constrains, by using regular expressions.

When construct the patterns you can with advantage use this homepage:
https://regex101.com/

Assignment 2: Test your validator class

Create a Unittest to test your five methods in your Validator class.

Assignment 3: Making experiments with finding, extracting and replacing text

Create a .Net Core Console application.

For the following text (a short description of H.C.Andersen 'The Emperors new clothes'):

"Two swindlers arrive at the capital city of an emperor who spends lavishly on clothing at the expense of state matters. Posing as weavers, they offer to supply him with magnificent clothes that are invisible to those who are stupid or incompetent. The emperor hires them, and they set up looms and go to work. A succession of officials, and then the emperor himself, visit them to check their progress. Each sees that the looms are empty but pretends otherwise to avoid being thought a fool. Finally, the weavers report that the emperor's suit is finished. They mime dressing him and he sets off in a procession before the whole city. The townsfolk uncomfortably go along with the pretense, not wanting to appear inept or stupid, until a child blurts out that the emperor is wearing nothing at all. The people then realize that everyone has been fooled. Although startled, the emperor continues the procession, walking more proudly than ever."

(source: https://en.wikipedia.org/wiki/The_Emperor%27s_New_Clothes)

1. Find the indexes of where the word emperor occur – with or without starting uppercase E.

2. Change the 'emperor' to 'teacher' and print out the new text.

3. Make capture groups of sentences, extract these and print them out