Day 02 .NET and C# Exercises

1) Methods with use of param
Make a new ConsoleApplication and a static method, Max, in the Program-class. It should be possible to call the method, Max, with a random number of double-parameter as shown in the code. Complete the declaration of Max.

```
class Program
{
  public static double Max(...)
  {
  }

  static void Main(string[] args)
  {
    double m1 = Max(28.5, 17.2);
    double m2 = Max(4.0, 10.8, 34.25, 2.0, 23.6);

    Console.WriteLine(m1+" "+m2);

    Console.ReadLine();
  }
}
```

The program prints `28.5 34.25`

2) String statistics with out parameters
Continue with the project from 1). Add a new method with a string parameter and two out parameters. The first out parameter should contain the number of words in the string and the second the longest word in the string. Example:

> If the input string is “Back to the Future”
> the out parameters should be 4 (number of words) and 6 (longest word, “Future”)

Test the method and write the to results.

3) extension methods
a) Create a Console application.

Create a new class file. Delete everything in the new class file except the using directives.
Copy the following to the file;

```
namespace ExtensionMath
{
  public static class MathExtension
  {
    public static int factorial(this int x)
    {
      if (x <= 1)
        return 1;
      else
        return x * factorial(x - 1);
      }
  }
}
```

It defines en extention method to the int class (remember even a value types can be considered as an object). Factorial compute the the factorial, e.i. 5! = 120.

Use the above to compute 5! and 7! And print the result.

b) Add another method in the MyMathExtension with the name, FixedDigits, which returns a string with the value of the int with the number of digits specified:

```
28.FixedDigits(5)         returns 00028

int n= -345;
n.FixedDigits(6);         returns -000345
```

Test your method.
Help: The string class has a PadLeft method you may use in your implementation.

c) Program you own extension method to the string class, isNumeric. The method should return true if the string contains a numeric and false otherwise. Use another namespace and another static class to define the method.

Test the method.

4) string, Char-methods and DateTime
Make a Console Application.
Define a Person class:
You may create a class-file in the WPF-project by right-clicking on the Console-project in the Solution Explorer and select Add | Class. Give the file an appropriated name, e.i. "myclasses.cs".

Add the following properties to the class

- `string CPR` (get og set)
- `DateTime birthDay` (get)

For CPR the set part must throw an Exception if the length being set is not 10 or any char in the string is not a digit.

You'll need the methods

- `string.SubString()`
- `Convert.ToInt32()` - The Convert-class which has a large number of methods to convert to the wanted type
- `char.IsDigit()` method

If you have time:
Declare your own Exception, IllegalCPRException, to throw when an illegal string is assigned to the CPR property.

Also: check if the first 6 char in the string assigned to CPR is a legal date. There are more solutions:

a) If you try to create a DateTime with a non-existing date an Exception is thrown.
b) Check the date explicitly: DateTime has methods IsLeapYear and DaysInMonth

Add a method to the Person class to split a string with a CPR number into 3 int out parameters: year (2 digits), month and day using 3 out parameters. The method returns a bool to indicate whether the splitting succeeded. (Handy but not very OOP like!)