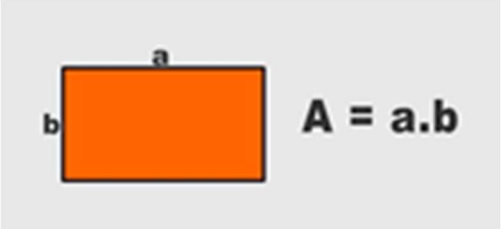
ESMAD | LTSIW | Programação Orientada a Objetos
Exercise Sheet nº2

Create a function to solve each of these points. For coding it is advisable to use Visual Studio Code:

1. **Function declaration**
    a. Create a function that displays the phrase "HELLO WORLD!" In an alert box. Give the function a good name. Run the function 3 times.

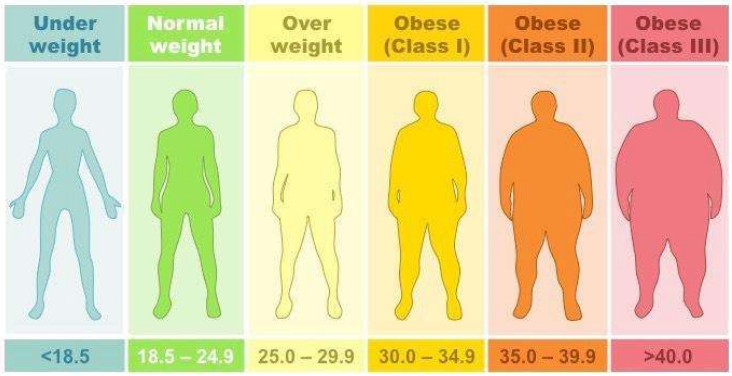
2. **Parameters**
    a. Create a function that prints "**HELLO [name]!**", Where **name** is a variable passed as a parameter. The **name** variable must be initialized "hard-coded".

    b. Create a function that prints the area of a rectangle when passed as parameters the length of the two sides.



    c. Create a function that displays the result of arithmetic operations in an alert box. The function must receive two integer values and an operator (+, -, *, /). All values must be obtained using **prompt** functions. Safeguard the division by 0 by presenting the user with a message stating that it is not possible to divide.

d. Create a function that displays the classification of a person's BMI (Body Mass Index) in an alert box using the respective weight and height as parameters. Both parameters must be collected using **prompt** functions.
BMI = Weight(kg) / [Height(m)]²



e. Create a function that simulates an echo. Given a string **s** and a number **n** it should print **s** written **n** times.

f. Create a function that starts by reading two integer values **a** and **b** given by the user, and that writes all integer values belonging to the range **[a, b[**

g. Create a function that calculates the sum of multiples of 3 existing in an interval **[a, b]**, where **a** and **b** are passed as parameters.

h. Create a function that returns the multiplication table for a given number passed as a parameter. If no number is passed, the function must print the multiplication table of 1.

1 X 1 = 1 1 X 2 = 2 1 X 3 = 3 1 X 4 = 4 1 X 5 = 5 1 X 6 = 6 1 X 7 = 7 1 X 8 = 8 1 X 9 = 9 1 X 10 = 10	2 X 1 = 2 2 X 2 = 4 2 X 3 = 6 2 X 4 = 8 2 X 5 = 10 2 X 6 = 12 2 X 7 = 14 2 X 8 = 16 2 X 9 = 18 2 X 10 = 20	3 X 1 = 3 3 X 2 = 6 3 X 3 = 9 3 X 4 = 12 3 X 5 = 15 3 X 6 = 18 3 X 7 = 21 3 X 8 = 24 3 X 9 = 27 3 X 10 = 30	4 X 1 = 4 4 X 2 = 8 4 X 3 = 12 4 X 4 = 16 4 X 5 = 20 4 X 6 = 24 4 X 7 = 28 4 X 8 = 32 4 X 9 = 36 4 X 10 = 40	5 X 1 = 5 5 X 2 = 10 5 X 3 = 15 5 X 4 = 20 5 X 5 = 25 5 X 6 = 30 5 X 7 = 35 5 X 8 = 40 5 X 9 = 45 5 X 10 = 50
6 X 1 = 6 6 X 2 = 12 6 X 3 = 18 6 X 4 = 24 6 X 5 = 30 6 X 6 = 36 6 X 7 = 42 6 X 8 = 48 6 X 9 = 54 6 X 10 = 60	7 X 1 = 7 7 X 2 = 14 7 X 3 = 21 7 X 4 = 28 7 X 5 = 35 7 X 6 = 42 7 X 7 = 49 7 X 8 = 56 7 X 9 = 63 7 X 10 = 70	8 X 1 = 8 8 X 2 = 16 8 X 3 = 24 8 X 4 = 32 8 X 5 = 40 8 X 6 = 48 8 X 7 = 56 8 X 8 = 64 8 X 9 = 72 8 X 10 = 80	9 X 1 = 9 9 X 2 = 18 9 X 3 = 27 9 X 4 = 36 9 X 5 = 45 9 X 6 = 54 9 X 7 = 63 9 X 8 = 72 9 X 9 = 81 9 X 10 = 90	10 X 1 = 10 10 X 2 = 20 10 X 3 = 30 10 X 4 = 40 10 X 5 = 50 10 X 6 = 60 10 X 7 = 70 10 X 8 = 80 10 X 9 = 90 10 X10 = 100

i. Create a function that adds **N** numbers passed by parameters, Use the **arguments** object to solve this problem.

j. Create a function that receives a child's first and last name and a set of strings that represent the name of each of your friends. The function must present the following

sentence: "**The [firstName] [lastName] has [nFriends] friends!**". Use the **Rest** parameter to resolve this issue.

3. **Function return**
   a. Create a **min(a, b)** function that returns the minimum of two numbers **a** and **b**.
   Test cases:

```
min(2, 5) == 2
min(3, -1) == -1
min(1, 1) == 1
```

   b. Create a **pow(x, n)** function that returns **x** at the power **n**. Or, in other words, multiply **x** by itself **n** times and return the result.
   Test cases:

```
pow(3, 2) = 3 * 3 = 9
pow(3, 3) = 3 * 3 * 3 = 27
pow(1, 100) = 1 * 1 * ...* 1 = 1
```

   Before calling the function, you must ask the user for **x** and **n**. The function must support only natural values of **n**: integers above 1.

   c. Create a function that checks whether a number passed as a parameter is prime or not. Remember that a prime number **N** is a natural number greater than 1 that has no divisors other than 1 and itself. Return true if it is prime and false otherwise. The function must be invoked within an **IF** structure and if it is true it must be presented "The number **[N]** is prime". Otherwise, "The number **[N]** is not prime".

   d. Create a function that returns the factorial of a positive integer value passed as a parameter. Examples: 0!=1; 1!=1; 5!=5x4x3x2x1= 120; 6!=6x5x4x3x2x1= 720;

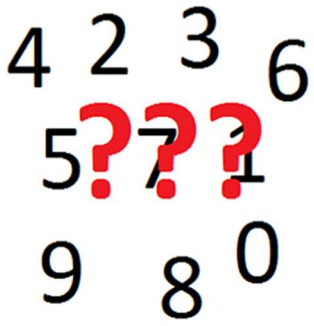
4. **Function expressions**

   a. Create a function expression called **isLeapYear** that should return **true** if a year passed by the user is a leap or **false** otherwise.

b. Create a function expression called **isPerfect** that checks if a number is perfect. A perfect number is a natural number for which the sum of all its own natural divisors (excluding itself) is equal to the number itself. For example, the number 28 is: 28=1+2+4+7+14. If the number is perfect return **true**. Otherwise, it must return **false**. Use the function in a loop that should continue to ask the user for a number until the number entered is perfect.

c. Create an abbreviated function expression (arrow function) that given a number from 100 to 999 check if it is a palindrome number. If so, it must return **true**. Otherwise, it must return **false**. Choose a good name for the function.

5. Create a guessing game. Start by generating a random number between 1 and 100. Then ask the user to guess the number. If the user enters a higher number, it should display the following text "**DOWN**". Otherwise, it must indicate: "**UP**". If the user gets it right, he should see the message **"CONGRATULATIONS, YOU GUESSED IT!"**.



In this game, you must have a function that takes two parameters: the number to guess and the user's attempt. The function must return:

-1 if the attempt is LOWER than the number initially generated
1 if the attempt is HIGHER than the number initially generated
0 if the attempt is EQUAL to the number initially generated

Make the game more interesting, and give the player just 5 tries. If the user reaches the limit, he should see: "PATIENCE, PLEASE PLAY AGAIN!".