TD4 Functions

Ex 4.1 Hello world!

Write a program that defines and uses:

1. A function, called **Hello()**, which simply displays "Hello world!" (it will have no parameters or return values).
2. A function, called **ManyHellos()**, which displays "Hello world!" a number of times equal to the value received in parameter and which returns no value, using the previous function.
3. A function, called **TooManyHellos()**, which does the same thing as **ManyHellos()** and returns 1 if it has displayed the message more than 10 times, 0 otherwise, using the previous function.

The function main() will use **TooManyHellos()** in a test statement which will display "That's a lot!" if more than 10 messages are displayed.

- *Note:* The result of **TooManyHellos()** should be considered as a Boolean value.

Ex 4.2 Digital base Conversion

Write a program that allows conversion between decimal and binary digital bases. This program must use separate functions to perform conversions. For that:

1. Write a function that takes a decimal number as input, converts it to binary, and then returns it.
2. Create a function that takes a binary number as input, converts it to a decimal, and returns it.
3. Test both functions in the main() function by displaying the result of the conversion.

Ex 4.3 Calculator

We want to write a program simulating a calculator.
We have three real numbers. According to a choice made from a menu displayed on the screen, we want to calculate the sum, the product, the average, the minimum, or the maximum of these three numbers. For that, write:

```
******* CALCULATOR*********
*  1 : ----------> Sum -----------------*
*  2 : ----------> Product -------------*
*  3 : ----------> Average ------------*
*  4 : ----------> Minimum ----------*
*  5 : ----------> Maximum ----------*
*  0 : ----------> Exit ----------------*
**************************
Enter your choice ?
```

1. A function that calculates the sum of three real numbers.
2. A function that calculates the average of three real numbers.
3. A function that returns the minimum between three real numbers.
4. A function that returns the maximum between three real numbers.
5. A function that displays the above menu:
6. A program that allows the user to use the calculator as many times as he want. The user must type 0 to exit.
7. A function that returns the minimum between six real numbers and then test it in the main() function.

Ex 4.4 Revise multiplication

We want to write a program that asks a user to recite his multiplication table.

The user starts by entering a number, then the program displays one by one the lines of the multiplication table of this number, leaving the result blank and waiting for the user to enter the result. If it is incorrect, an error message is displayed with the correct value. Then we go to the next line.

If at the end all the answers are correct, a congratulatory message is displayed, otherwise the number of errors made is displayed.

A possible execution is shown here (user inputs are displayed in italics).

REVISION OF MULTIPLICATION TABLES

Would you like to revise multiplication by: 7
1 x 7 = *7*
2 x 7 = *14*
3 x 7 = *28*
Error! 3 x 7 = 21 not 28
4 x 7 = *28*
5 x 7 = *36*
Error! 5 x 7 = 35 not 36
6 x 7 = *42*
7 x 7 = *49*
8 x 7 = *56*
9 x 7 = *63*
10 x 7 = *70*
You've made 2 mistakes!
You haven't yet learnt your multiplication table by 7.

For this, we'll write:

1. A function **read_n()** that reads the value of the number whose multiplication table the user wants to revise.

   This number should be between MIN=2 and MAX=9 (MIN and MAX being constants defined at the start of the program).

   If the number is incorrect, the function asks for an integer again. It will return the entered value once it is valid.

2. A function **one_multiplication()** which displays a line in the table and retrieves the user's input, then checks if it is correct.

   It will take as parameters the numbers to be multiplied.

   The return value will indicate whether the value entered is correct.

3. A function **review_multiplications()** that calls the previous function several times to review the multiplications for each line of the multiplication table for the number passed to it as a parameter.

   It will return the number of errors made by the user.

4. A function **main()** uses these three functions and then displays the result (congratulations or number of errors made).

5. Add to the program a function **Square()** that allows you to revise the squares of the integers within the range given by the user.

   It will take the bounds of the interval as parameters and will have to use the **one_multiplication** function from question 2.

   A menu is used to select the revision, as shown in the example beside.

*******REVISION OF MULTIPLICATIONS******
*----------
* 1- Table revision
* 2- Square revision
* 0- Exit
*----------
Your choice? *2*
Square revision:
From *2* **to** *6* **:**
2 x 2 = *4*
3 x 3 = *9*
4 x 4 = *16*
5 x 5 = *25*
6 x 6 = *36*
Well done! You know all your squares from 2^2 to 6^2
…(redisplay menu...)..etc