Practice 5: Recursion (1 session)

Week 4th November 2024

Exercise 1

Write a recursive module in C that prints out a right triangle on the screen based on the height received as a parameter. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

Example of operation

```
$ ./exercise1

Introduce the triangle height: 5
*
**
***
****
*****
```

Exercise 2

Write a recursive module in C that receives an integer and counts the number of digits it has. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

Example of operation

```
$ ./exercise2

Introduce an integer number: 5
# digits: 1

$ ./exercise2

Introduce an integer number: 5637
# digits: 4
```

Exercise 3

Write a recursive module in C that receives two integers n and m, and returns n power to m. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

```
$ ./exercise3

Introduce two integer numbers n and m: 2 3
The result of 2^3 is: 8

$ ./exercise3

Introduce two integer numbers n and m: 2 4
```

```
The result of 2^4 is: 16
```

Exercise 4

Write a recursive module in C that receives an integer and returns the number of even digits and the number of odd digits it has. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

```
$ ./exercise4

Introduce an integer number: 34567
# even digits: 2
# odd digits: 3

$ ./exercise4

Introduce an integer number: 22344
# even digits: 4
# odd digits: 1
```

Exercise 5

Write a recursive module in C that receives an integer and checks if it is binary (only has digits 0 and 1), returning `true` if it is and `false` if it is not. Note: It is necessary to implement a C program to test the module (it will be called from the `main()`).

```
$ ./exercise5

Introduce a binary number: 0101010
The number is binary

$ ./exercise5

Introduce a binary number: 010021
The number is NOT binary
```