NOMBRE		ETAPA / CICLO	CURSO
		CFGS DAW/DAM	1º
APELLIDOS		ASIGNATURA/MÓDULO	CONVOCATORIA
		PROGRAMACIÓN	ORDINARIA
DNI	FECHA	NOTA	
	28-11-2022		

1. The aim of this exercise is to manage a matrix of Strings (String[][]).
   a. **(0,40p)** Create a java class having a function to **fill** a two-dimension array of Strings. The row and column size will be requested to the user. Content of each element will be the row number followed by the column number.
   b. **(0,40p)** Add a function to **print** the matrix as showed in part a of this exercise.
   c. **(0,40p)** Add a function to return the **sum** of all numbers in the vector
   d. **(0,40p)** Add a function to return the **average**
   e. **(0,40p)** Create the **main function** to show the result which should be like:

```
Give me the number of rows:
5
Give me the number of columns:
4
00 01 02 03
10 11 12 13
20 21 22 23
30 31 32 33
40 41 42 43
The sum is: 430
The average is: 21.50
```

2. For this exercise, a matrix of integers (two-dimensional array) is used.
   a. **(0,75p)** Create a function to **count all even** elements of the matrix.
   b. **(0,75p)** Create a function to **return the even number and the position of all even elements**.
   c. **(0,50p)** Create the class and the main function to display an output like the one shown below.

```
1 3 0
5 3 5
4 9 6
8 7 4

The count of all even elements is: 5
There even number 0 is in the row 0 col 2
There even number 4 is in the row 2 col 0
There even number 6 is in the row 2 col 2
There even number 8 is in the row 3 col 0
There even number 4 is in the row 3 col 2
```

3. Binary search.
   a. **(1,00p)** Create a function to **sort** a two-dimensional array of persons with their height (see the annex) and sort it by the name. Remember to also implement the **swap** function. Use this header:

```
public static void bubble(String[][] array)
```

   b. **(1,00p)** Create an **efficient** function to find a person by his/her name, just the first person found, in a **sorted array**. You can solve this exercise recursively (you can use a function helper) or iteratively:

```
private static String search(String[][] array, float height)
```

   Expected result:

```
Person's name: Carl
Height found: 1,62
```

4. **(2,00p)** Create a program with at least three functions. One function will be the **main function**. Another one will be a function to **fill** an integer matrix with **random numbers** within a **requested range** (the lower and upper limit will be requested to the user). The **dimension** of the matrix (number of rows and columns) will be requested to the user too. And finally, a function to fill a vector with all even numbers from the matrix **without repetition**.

```
Give me the number of rows:
5
Give me the number of columns:
4
Give me the lower random limit:
0
Give me the upper random limit:
73

Example of the expected array:
[{25,13,11,73},
{10,11,12,12},
{15,4,10,30},
{30,55,69,33},
{40,41,49,40}]

The array returned will contain: [10, 12, 4, 30, 40]
```

5. Choose only one of the following exercises and solve it **efficiently** twice. First recursively **(1,00p)** and finally iteratively **(1,00p)**:
    a. Given a non-negative int n, compute the count of the occurrences of 7 as a digit, except that a 7 with another 7 immediately to its left counts double, so 7717 yields 4. Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

        count7(7) → 1
        count7(717) → 2
        count7(7717) → 4

    b. Given two strings, compare lexicographically two strings ignoring case differences. Return -1 if the first string lexicographically precedes the second string, 1 if the second string lexicographically precedes the first string and 0 if the first string is equals the second string. <u>You can't use the functions compareTo and compareToIgnoreCase.</u>

        compare("plane","zoo") → -1
        compare("zoo","plane") → 1
        compare("zoo","zoo") → 0

Annex

Bubble sort algorithm for sorting integer arrays

```
public static void bubble(int[] array) {
    int i, j;
    for (i = 1; i < array.length; i++)
        for(j = 0; j < array.length - 1; j++)
            if(array[j]>array[j+1])
                swap(array, j, j + 1);
}
```

Heights

```
String[][] persons = {
            {"Peter","1.85"},
            {"John","1.86"},
            {"Thomas","1.78"},
            {"Carl","1.62"},
            {"Lewis","1.68"},
            {"Michael","1.65"},
            {"Vanessa","1.70"},
            {"Anne","1.69"}
};
```