NOMBRE		ETAPA / CICLO	CURSO
		CFGS DAW/DAM	1º
APELLIDOS		ASIGNATURA/MÓDULO	CONVOCATORIA
		PROGRAMACIÓN	ORDINARIA
DNI	FECHA	NOTA	
	22-11-2021		

1.

   a. **(0,50 p.)** Create a program that fills in a two-dimensional vector, of size requested from the user, as follow:

```
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
```

   b. **(0,25 p.)** Add a function to print the vector as showed in exercise one.
   c. **(0,50 p.)** Add a function that returns the sum of all number in the vector and create the main function to show the result as you can see in the exercise one.

2.

   a. **(1, p.)** Create a function to count all even elements of the matrix. The multidimensional array is formed by Strings.
   b. **(1,75 p.)** Create a function that returns the position of all even elements. Use this function to show the positions. Create the main function to display an output like the one shown below.

```
1 3 0

5 3 5

4 2 6

8 7 4


The count of all even elements is: 6

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 2 is in the row 2 col 1
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
```

Hint: You have to use the function created in the exercise *a* to solve the exercise *b*.

3.
   a. **(1,25 p.)** Create a function to sort a two-dimensional array of persons with their height (see the annex) and sort it by height. Use this header:

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

   b. **(1,25 p.)** Create an **efficient** function to find the first person with a certain height 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)
```

```
Person height to search:

1,62

Search: Carl
```

4. **(1,25p)** Create a function that takes an array of two-dimensional integers and returns a one-dimensional array with the same integers but no repetitions. You can use any helper function you need. The size of the array returned must be just enough to store the integers. That is, its size must be exactly what is necessary to store the non-repeating integers.

```
public static int[] withoutRepetitions(int[][] array)
```

```
int[][] array = {{1, 4},
                         {1, 4},
                         {2, 4},
                         {1, 4},
                         {1, 7},
                         {9, 4},
                         {3, 3},
                         {0, 4},
                         {1, 4}};


Array without repetitions:

1 4 2 7 9 3 0
```

5. Choose only one of the next exercises, and solve **efficiently** it twice. First recursively **(1,25p)** and finally iteratively **(1 p)**:

a. Given a string, return recursively a "cleaned" string where adjacent chars that are the same have been reduced to a single char. So "yyzzza" yields "yza", and "abbbcdd" yields "abcd".

```
stringClean("yyzzza") → "yza"

stringClean("abbbcdd") → "abcd"

stringClean("Hello") → "Helo"
```

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][1]>array[j+1][1])
                swap(array, j, j + 1);
}
```

Heights

```
String[][] heights = {

            {"Peter","1.85"},
            {"John","1.86"},
            {"Thomas","1.78"},
            {"Carl","1.62"},
            {"Lewis","1.68"},
            {"Michael","1.65"},
            {"Vanessa","1.70"},
            {"Anne","1.69"}

};
```