# SOFTWARE 2 PRACTICAL

ARRAYS & STATIC METHODS

Week 3 – Practical 2

For this week practical, you should create a Java project (see last week introduction to VS Code), and write all your classes in the **src** folder.

## Exercise 1:

Write a **static method** `int toBase10(String binary)` that take a String representation of a binary number (base 2), convert it into a decimal number (base 10) and return the base 10 value. To compute such a value, we need to understand what a binary number is.

| **Index** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **Binary** | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **Decimal 139** | $1 \times 2^7$ | $0 \times 2^6$ | $0 \times 2^5$ | $0 \times 2^4$ | $1 \times 2^3$ | $0 \times 2^2$ | $1 \times 2^1$ | $1 \times 2^0$ |
| | 128 | 0 | 0 | 0 | 8 | 0 | 2 | 1 |

The binary number 10001011 represents the number 139, whereas the number 11111111 represents 255.

## Exercise 2: *reinventing the wheel! (again)*

For this question we are emulating the method `split()` from the type `str` In Python. Create a class `TextUtils` and implement the static method `String[] split(String text)` where `text` is a string. The method returns an array of String which contains the words from the text (split by a blank space).

You must NOT use the any existing classes such as `StringTokenizer` to solve the problem.

## Exercise 3: *a more flexible* `split`.

In `TextUtils`, overload the method `split(String text, String separators)` where `text` is a string to be split, `separators` is a string containing all the characters used to split the text (for example `",.!? "`). The method returns an array of String containing the list of tokens separated by one of the separators.

## Exercise 4:

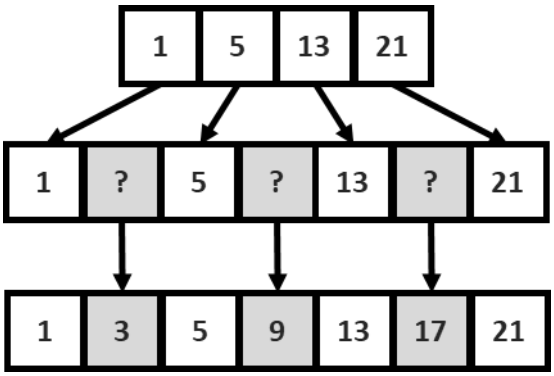
Write a static method `rasterise(int[] data, int width)` that transforms a 1D array passed as parameter into a 2D array, where each sub-array have `width` elements. If the length of the 1D array is not a multiple of `width`, the method should return `null`.

For example:

```
rasterise({1,2,3,4,5,6,7,8},4)→ {{1,2,3,4},{5,6,7,8}}
rasterise({1,2,3,4,5,6,7,8},3)→ null
```

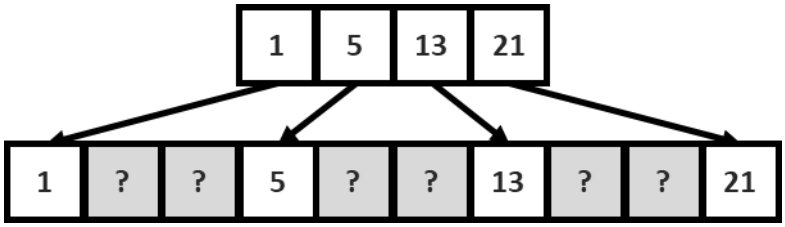
## Problem:

The aim of this problem is to resample a set of data points store in an array. Write a class `LinearInterpolation` which will contains a set of static methods to resample data point using linear interpolation. To start with we will look at the simple case where we want to (almost) double the number of sample point using linear interpolation. For example, given an array of 4 known values (shown below), we want to resample the array in order to have 7 values. A simple approach is to use linear interpolation (see linear interpolation on Wikipedia). To compute the values in the grey boxes, you just have to take the two neighbouring boxes, add them together and then divide by 2.



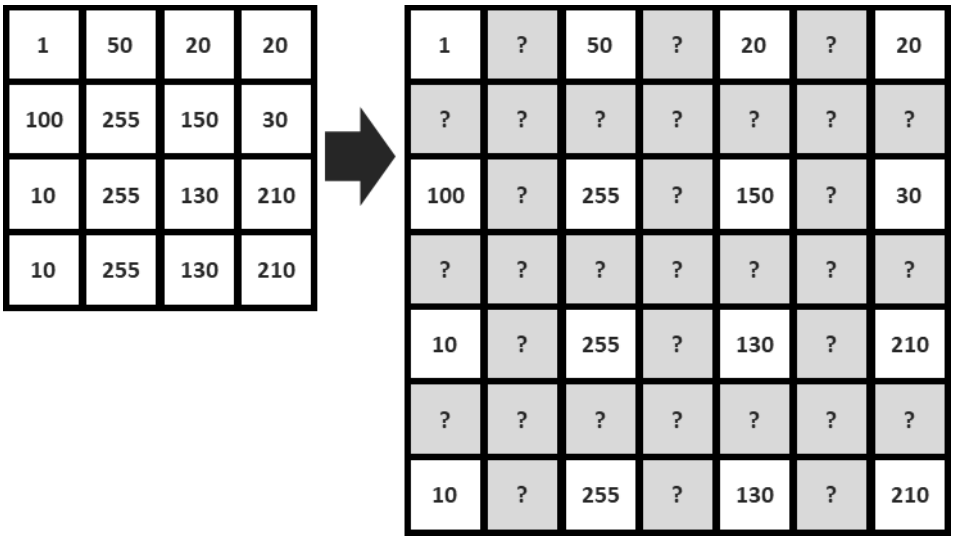

1. Implement a static method `int[] resample(int[] datapoints)` that does just that.

Something more challenging is to be able to resample the data for any factor (`int`). Below is an example of a factor of three.




2. Implement the method `int[] resample(int[] datapoints, int scale)` that does just that. You will need to understand the formulae given in the section "Linear interpolation between two known points" on the Wikipedia page. In our case the x are the index of the element in the array, and the y are the value stored in the array.

Bilinear interpolation can be used in image processing to resample an image (scaling up the resolution of an image). In this case, the image can be seen as a 2D array of pixel values. Considering a black & white image (a.k.a. greyscale image), the "colours" range from 0 (black) to 255 (white). For example, to double the size of the $4 \times 4$ image below, we first build a $7 \times 7$ array with missing values. To compute the missing values, we first deal with the even rows (0, 2, 4, 6). When considering the row ***r=0***, we can use the linear interpolation seen in 1 to complete that row. Then we proceed to the row ***r=r+2*** and so on. Once the rows 0, 2, 4 and 6 have been interpolated, we tackle the columns 0, 1, …, 7. We interpolate one column at a time using the method seen in 1.




3. Implement `int[][] resample(int[][] image)` which double the size of a greyscale image.

4. Implement `int[][] resample(int[][] image, int scale)` which resample a greyscale image given a scale factor.

5. More challenging, but also more useful would be to implement a resampling method where the scaling factor may not be a whole number. Implement this functionality with the static method `int[][] resample(int[][] image, double scale)`.