For this assignment, you are asked to write a function capable of reading a CSV file and printing it to the console in a readable format. However, we will add an additional constraint: you are prohibited from using for and while loops. So, you will have to use the functions map, filter and reduce views in class.

Recall that the CSV file format is a convenient format for storing data represented in the form of a two-dimensional matrix. However, this format is not human readable. As the following example demonstrates, the elements of the same column of a CSV file are not aligned, making it difficult to read.

Surname, First name, Assignment #1, Assignment #2, Assignment #3
Deutsch,David,100,100,100
Diaz,Manual,85,99,78
Slimani,Hakim,76,83,91
For this reason, the contents of a CSV file are frequently formatted into a table made up of ASCII characters. Here is the representation of the previous CSV in a rectangular table.

```
+-------+------+---------+---------+---------+
|    Nom|Prenom|Devoir #1|Devoir #2|Devoir #3|
+-------+------+---------+---------+---------+
|Deutsch| David|      100|      100|      100|
+-------+------+---------+---------+---------+
|   Diaz|Manuel|       85|       99|       78|
+-------+------+---------+---------+---------+
|Slimani| Hakim|       76|       83|       91|
+-------+------+---------+---------+---------+
```

You must implement the function grilleCSV taking as parameter the name of a CSV file and returning a text representing a table whose format is specified below. You can assume that each row in the CSV file contains the same number of items.

In order to align the elements of the same column, spaces are added to the left of the shorter elements. For example, in the first column of the table above, the longest element is "Deutsch" with seven characters. Thus, spaces are added to the left of each element in order to obtain boxes of seven characters in width for this column.

To aid readability, the boxes are separated by lines composed of the characters "-", "|" and "+", used respectively to represent horizontal lines, vertical lines and the intersections of two lines.

Newline "\n" characters are used to separate each line of text. However, there should not be a blank row after your table. So be careful not to add a line break after your table.

In addition to adhering to the specification, your code must not contain any for or while loops. In order to help you get started, you can reuse the following version of readCSV that does not use loops.

```
def readCSV(path):
    lines = readFile(path).split('\n')
    if lines[-1] == '': lines.pop()
    return list(map(lambda line: line.split(','), lines))
```

You can also use the column function below demonstrating how it is possible to retrieve a specific column in a matrix using map.

```
def column(matrix, i):
    return list(map(lambda row: row[i], matrix))
```

Once your program retrieves a column, it is possible to use map and reduce on that column. For example, given a column ["a", "bb", "ccc"], the expression *list (map(len, ['a','bb','ccc']))* will return a list containing the length of each item in that column.

You need to break your program down into auxiliary functions. You will be evaluated on your choice of functional decomposition. As usual, your code should contain explanatory comments and meaningful camelCase identifiers.

Eventually, you need to write a function that runs unit tests. It is not possible to write unit tests for the function gridCSV because it reads a file which will not necessarily exist on the proofreader's computer, but you must write tests for the auxiliary functions that you deem relevant to test.