Project 1: **Arithmetic Formatter**

Create a function *arithmeticArranger()* that receives a list of strings that are arithmetic problems and returns the problems arranged vertically and side-by-side. The function should optionally take a second argument. When the second argument is set to **True**, the answers should be displayed.

For example

Function Call:

```
arithmeticArranger({"32 + 698", "3801 - 2", "45 + 43", "123 + 49"})
```

Output:

```
    32      3801      45      123
+ 698      -  2     +43      +49
-----    ------    ----    -----
```

Function Call:

```
arithmeticArranger({"32 + 8", "1 - 3801", "9999 + 9999", "523 - 49"}, True)
```

Output:

```
  32         1       9999      523
  +8    - 3801     + 9999      -49
----    ------     ------    -----
  40     -3800      19998      474
```

Rules

The function will print the correct conversion if the supplied problems are properly formatted, otherwise, it will **return** a **string** that describes an error that is meaningful to the user.

Situations that will return an error:

- If there are **too many problems** supplied to the function. The limit is **five**, anything more will return: **Error: Too many problems.**
- The appropriate operators the function will accept are **addition** and **subtraction**. Multiplication and division will return an error. Other operators not mentioned in this bullet point will not need to be tested. The error returned will be: **Error: Operator must be '+' or '-'.**
- Each number (operand) should only contain digits. Otherwise, the function will return: **Error: Numbers must only contain digits.**
- Each operand (aka number on each side of the operator) has a max of four digits in width. Otherwise, the error string returned will be: **Error: Numbers cannot be more than four digits.**

If the user supplied the correct format of problems, you will return with these rules:

- There should be a single space between the operator and the longest of the two operands, the operator will be on the same line as the second operand, both operands will be in the same order as provided (the first will be the top one and the second will be the bottom.
- Numbers should be right-aligned.
- There should be four spaces between each problem.
- There should be dashes at the bottom of each problem. The dashes should run along the entire length of each problem individually. (The example above shows what this should look like.)

Development

Write your code in a class with your Full Name, the first character of ever name should begin with an upper-case letter and the other characters in lower case, ex. If your name is RAMAN J HALMAT, your class should be named RamanJHalmat. The main function should just contain the function call with multiple test cases as the arguments and nothing else, no main function is also accepted.