

We will create a simple string calculator with the following signature:

```
fun add(numbers: String): Int
```

This is a TDD (test driven development) exercise.
So please keep in mind: Test(s) first!

Any questions so far? 😊

1. The method can take up to two numbers, separated by commas and will return their sum.

For example:

`add("")` should return 0

`add("1")` should return 1

`add("1,2")` should return 3

2. Allow the add method to handle an unknown amount of numbers.

For example:

`add("1,2,3,4,5,6")` should return `21`

3. Allow the add method to handle new lines between numbers.

For example:

`add("1\n2,3")` should return `6`

You do not need to cover something like: `"1,\n2,3"`

4. Support different single character delimiters which will be given in the following format:

`"//delimiter\nnumbers"`

For example:

`add("//*\n1,2\n3*4")` should return 10

`add("//]\n1\n2]3")` should return 6

5. Calling add with a negative number will throw an exception "negatives not allowed" plus the negative that was passed. If there are multiple negatives, show all of them in the exception message.

For example:

`add("-3,-8")` should throw `Exception("negatives not allowed: -3, -8")`

6. Numbers bigger than 1000 should be ignored.
For example:

`add("2,1001")` should return 2

7. Delimiters can be of any length with the following format:

`"//[delimiter]\nnumbers"`

For example:

`add("//[***]\n1***2,3")` should return 6

`add("//[Hello]\n1,2Hello3")` should return 6

8. You can have multiple delimiters:
`"//[delimiter1][delimiter2]\nnumbers"`

For example:

`add("//[*][%]\n1*2,3%4")` should return `10`

8. Make sure you can also handle multiple delimiters with length longer than one char:

`"//[delimiter1][delimiter2]\nnumbers"`

For example:

`add("//[***][%%%]\n1***2,3%%%4")` should return `10`