Exercise on Test Driven Development

String Calculator

- Before you start:
  - Try not to read ahead
  - Do one task at a time
  - Work incrementally
  - Make sure you only test for correct inputs, there is no need to test for invalid inputs

- Exercise by Roy Osherove

String Calculator

1. Create a simple string calculator with a static method **`int Add(string numbers)`**
   a. The method can take 0, 1 or 2 numbers, and will return their sum (for an empty string it will return 0) for example "" or "1" or "1,2"
   b. Start with the simplest test case of an empty string and move to one and two numbers
   c. Remember to solve things as simply as possible so that you force yourself to write tests you did not think about
   d. Remember to refactor after each passing test
2. Allow the **`Add`** method to handle an unknown amount of numbers
3. Allow the **`Add`** method to handle new lines between numbers (instead of commas).
   a. the following input is ok: "1\n2,3" (will equal 6)
   b. the following input is NOT ok: "1,\n" (not need to prove it - just clarifying)

String Calculator

6. Support different delimiters
   a. to change a delimiter, the beginning of the string will contain a separate line that looks like this: "//[delimiter]\n[numbers…]" for example "//;\n1;2" should return three where the default delimiter is ';' .
   b. the first line is optional. all existing scenarios should still be supported
7. Calling **`Add`** with a negative number will throw an exception "negatives not allowed" - and the negative that was passed. if there are multiple negatives, show all of them in the exception message
8. Numbers bigger than 1000 should be ignored, so adding 2 + 1001 = 2
9. Delimiters can be of any length with the following format: "//[delimiter]\n" for example: "//[***]\n1***2***3" should return 6
10. Allow multiple delimiters like this: "//[delim1][delim2]\n" for example "//[*][%]\n1*2%3" should return 6.
11. Make sure you can also handle multiple delimiters with length longer than one char