Instructions

Second step of the interview process is completing a small assignment so you have the chance to really show off what you can do and how you think about organizing your code. The main goal is to assess problem solving skills, code organization, and testing. Technology know-how is a lesser factor.

Ground rules

- Feel free to use the internet, documentation, books or other literature for research and referencing
- Please DO NOT copy and paste the code from tutorials or StackOverflow
- The code MUST BE your original work, and you should be able to explain how and why it works (or does not); also, explain other considerations you've been thinking about.
- You can ask us as many questions as you need while working on the task via email
- We value well crafted code which is easy to understand
  It's always better to complete only one task with well designed code, than to complete all tasks without paying too much attention to the code quality.

Technology and tools to use

- Visual Studio
- C# programming language
- xUnit library for writing and running automated tests
- Expectations:
- Please comply with the deadline for submission of the tasks that has been communicated
- For any questions or uncertainty please send questions to muhamed.isein@haselt.com, bojan.veljanovski@haselt.com , stefan.georgievski@haselt.com
- Once done, the answers should be sent via email, also the codebase should be sent in a ZIP file attachment as well (without obj, bin, and packages directories/those are ignored), alongside your comments/description of your thinking process while you were solving the problem, such as: How did you start with solving it?; Any surprises during the process? What was the most challenging thing to understand? What was the most challenging part to solve?

Let's begin!

TASK 1: Text Calculator

First step: Create a method “add” that takes a string (of numbers) and returns a string (of their sum):

```
string Add(string number)
```

- The method can take 0, 1 or 2 numbers separated by commas and return their sum
- an empty string will return "0"
- Example of inputs :
  - ""
  - "1"
  - “5,6”
  - “1.1,2.5”
- Example of outputs:
  - "0"
  - "1"
  - "11"
  - "3.6"

Many numbers
Allow the method to handle an unknown number of arguments, not just 3 numbers.

Missing number in last position

- "1,3," is invalid input and the method should throw InvalidOperationException with the message "Missing number in last position."

Negative numbers
Calling the method with negative numbers will throw an exception.

- "-1,2" is invalid input and throw exception with message "Negative not allowed : -1"
- "2,-4,-5" is invalid input and throw exception with message "Negative not allowed : -4, -5"

Desired outcome

- Implemented "TextCalculator" class with "Add" method that implements all above specified rules
- Test and proof that all cases work as specified by writing test cases, for each rule. Also, try to find edge cases in order to make the solution robust. Name the test class "TextCalculatorTests.cs" and write all test code there using xUnit.

HASELT

TASK 2

Create the simplest API .NET Core project that exposes TASK 1’s functionality over an API endpoint.
Create the simplest web frontend application that provides an UI for the user to do text calculations. Integrate the API.
Write a README, to explain how to run this program locally - so other developers can use it.