C# Coding Test

In this coding test you will have the ability to show one of two abilities. You will be able to show us your current ability to write C# or if you can build yourself up to the level required for this test in good time.

This test is split into three basic parts, one for writing the basic code, another part for automated tests and finally your ability to use GitHub for source control.

You may complete as much of this test as you see fit, this is just to give us an understanding of your current knowledge or how fast you can learn. Please feel free to use Google if you get stuck, this is a test of you ability to self-learn as well.

If you get stuck and cannot complete part of the task please still send us your code. Just because you cannot finish the test doesn't mean that you don't have a chance to show us your abilities. If you cannot upload to GitHub for whatever reason please send your code in a compressed format to the email at the end of this document.

Prerequisites

You will need the following tools to complete this test:

- Visual Studio 2019 Community Edition
- GitHub Account (Free)

Synopsis

This will be a class library either written for .NET Framework or .NET Core (your choice) which will analyse strings provided. The goal is to output the most frequent words in a string, the word that is used most. Below is an example string:

"The sun shines over the lake"

For the above string the most frequent word would be "the" as it appears twice in the string. The following are assumptions and definitions that limit the scope of the program:

Word: A word is a string of text the is limited to the characters a-z and A-Z. For example "ABCdef" would be a word.

Letter Case: When counting how often a word appears in a string, we do not want to worry about case. For example the string above "The sun shines over the lake" would count "the" two times regardless if the input to the function is "THE", "tHE", "the" or any other combination of case.

Input Text: You should check to make sure that the text only contains characters that meets the requirements of a word (a-z) and (A-Z). You should throw an exception should the string be invalid. The words should only be split by a space character.

1) Implementation

Create a class in your class library that implements the following interface:

```
namespace Test
{
    public interface IWordFrequency
    {

        string Word { get; set; }
        int Frequency { get; set; }

    }
}
```

Now create a class which implements the following interface:

```
using System.Collections.Generic;

namespace Test
{
    public interface IWordFrequencyAnalyzer
    {

        /// <summary>
        /// Which word appears the most in the string
        /// </summary>
        /// <param name="text">The string of string you are testing against</param>
        /// <returns>How often the most frequent word appeared</returns>
        int CalculateHighestFrequency(string text);

        /// <summary>
        /// How often does a certain word appear in a string
        /// </summary>
        /// <param name="text">The string of string you are testing against</param>
        /// <param name="word">The word you wish toe count the frequency of</param>
        /// <returns>How often the word provided was found in the text
provided</returns>
        int CalculateFrequencyForWord(string text, string word);

        /// <summary>
        /// The most frequent words, return amount defined by the number parameter
        /// </summary>
        /// <param name="text">The string of string you are testing against</param>
        /// <param name="number">How many words you wish to return</param>
        /// <returns>A list of the implementation of the IWordFrequency interface of
the top most common words</returns>
        IList<IWordFrequency> CalculateMostFrequentWords(string text, int number);

    }
}
```

Implement the three methods defined in this interface:

- CalculateHighestFrequency should return the highest frequency in the text (several words might actually have this frequency)
- CalculateFrequencyForWord should return the frequency of the specified word
- CalculateMostFrequentNWords should return a list of the most frequent “n” words in the input text, all the words returned in lower case. If several words have the same frequency, this method should return them in ascendant alphabetical order (for input text “The sun shines over the lake” and n = 3, it should return the list { (“the”, 2), (“lake”, 1), (“over”, 1) }

2) Test Cases

Implement some automated test cases to check that the provided methods work as expected. You may use any third party framework for this, some examples are xUnit, NUnit and MSTest.

3) GitHub

Create a public repository on GitHub, commit your code to this repo and put any comments you wish to make in the ReadMe.md file. We will let you know once we have the code so that you may remove the repository should you wish.

Completion

Once complete please email a link to your GitHub repository to richard.jarrett@bluerocksystems.co.uk and steve.batchelor@bluerocksystems.co.uk, please make sure richard.jarrett@bluerocksystems.co.uk will be able to access the repository.