Assignment 4

Deadline:	Submit online by midnight Thursday, 7th of May 2015
Evaluation:	10% of your final grade (10 marks) Internal students must demonstrate their program to a tutor in order to receive a mark.
Late Submission:	10% off per day late
Work	This assignment is to be done **individually** – your submission may be checked for plagiarism against other assignments and against Internet repositories. If you adapt material from the Internet, then acknowledge your source.
Purpose:	Learn to: work with file I/O, use .NET data structures, devise unit tests and develop software using TDD.

Problem to solve:

Design an application that opens and analyses files containing text.

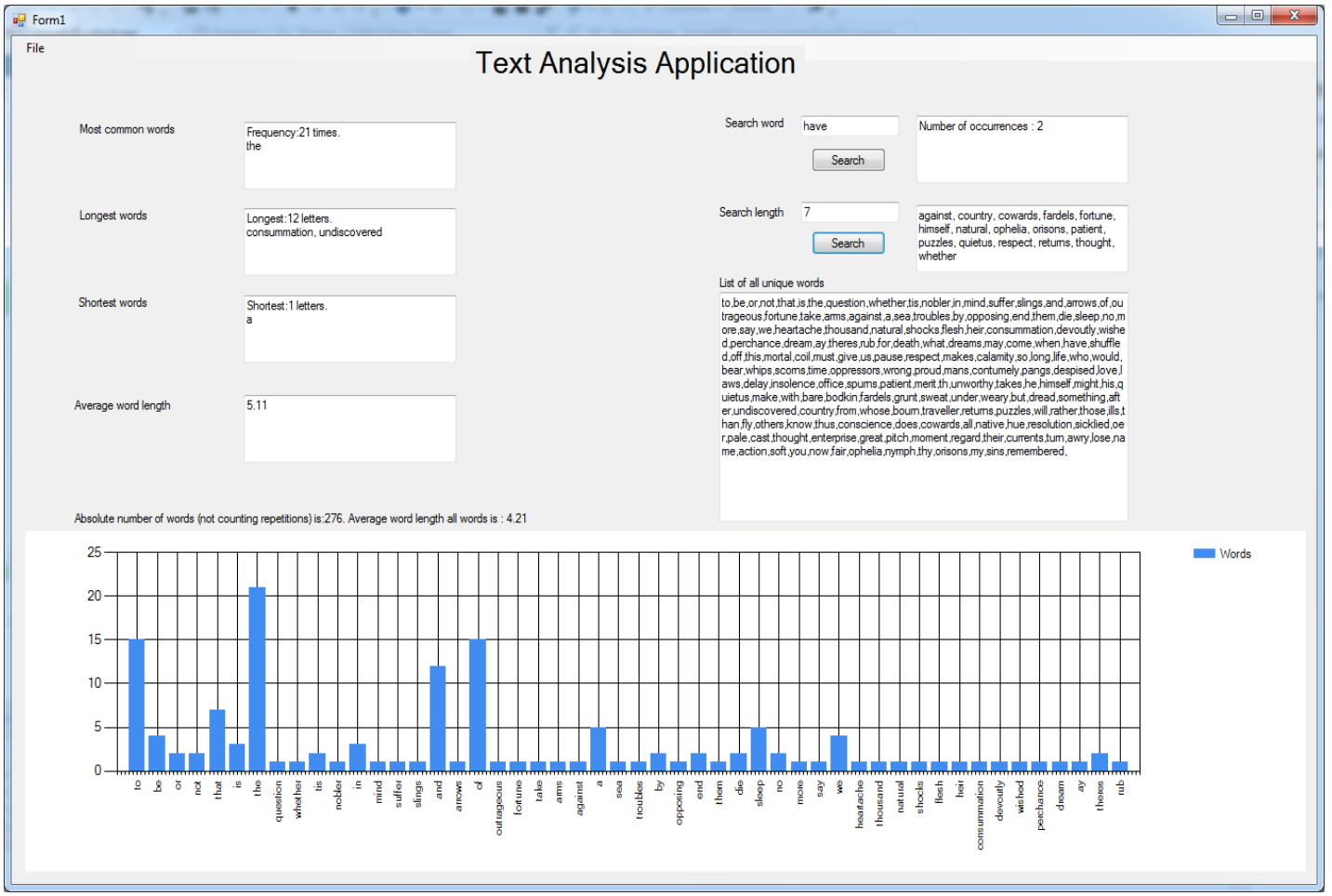
Functional Requirements:

Create an application that analyses text documents. It should open a text file, read each word into an appropriate data structure and analyse them. Once the contents of the document has been read and analysed, some statistics and figures about the contents should be displayed.

These statistics should be:

The most common word	(including the number of times it occurs in the file)
The longest word	(including the length of the word)
The shortest word	(including the length of the word)
The average word length	(average of all the unique words)

The image below is a screenshot of what your application could look like and what the output should be given a test file 'assig4EXAMPLE.txt'.



For each metric, the word(s) that fulfil the metric should be displayed. From the example above, if the longest word criterion is met by more than one word ('consummation', 'undiscovered'), then they should both be outputted together with their length.

The application should also include functionality to search for specific words. If the user searches for a specific word, the number of times that word appears in the text file should be displayed. If a specific word length is searched for, then all words of that length should be displayed (in a sorted order).

A list of all the words (without repetitions) should be displayed in an appropriate form control.

Also, you are to include a chart in your application that plots a histogram of at least 50 words from the text file, displaying the number of occurrences of each word in the text.

You should decide what controls to use in the design of your application. However, it should include a Menu and use a dialog box that allows the user to select a file to open (.txt files only).

You can assume that there will be no numerical characters or apostrophes in the test file used for your assignment. If your assignment implementation can handle combinations of multiple whitespace characters and standard punctuation, then an **extra 0.5 of a mark** will be awarded to you if you have been deducted points for anything else (please indicate in your comments if your application has this capability.). Otherwise, make sure that your assignment works on text files containing single spaces and new line characters to separate the words (eg. 'assig4EXAMPLE.txt').

Once you have developed a working program, refactor it in order to make it even better and thus gain maximum marks.

The above application must be developed using the TDD approach. The unit tests are expected to perform near 100% code coverage for user-defined methods (this excludes event-handlers).

Marking criteria:

Your application will be marked on functionality and the usage of good software engineering principles discussed in class, including the use of the TDD approach.

Functionality (3 marks)

- The display of appropriate statistics (2% - 0.5% for each metric)
    - deductions may be made for displaying metrics that do not fulfil the requirements.
- Search functionality (1%)
    - Search by word and appropriate display of search result (0.5%)
    - Search by word length and appropriate display of search result (0.5%)
- Appropriate use of controls
    - the use of menu, dialog box(es), and other controls as appropriate (1%)

Use of appropriate data structure (1.5%)
The data structures that have been taught in the lectures so far should be used appropriately.

Proper Error Handling Mechanisms (1.5%)
The error handling mechanisms that have been taught in the class, including if/else statements, throw statements, and try/catch/finally statements must be used appropriately.

Use of TDD (2%)
The application should be developed using TDD approach. Unit tests must be written for user defined methods. Comprehensive (near 100% code coverage) unit testing is expected.

Code Refactoring (1%)
Incorporating refactoring in the development of the code – to be demonstrated by including in the zipped file of the original .cs or .vb file before it was refactored (**distance students** – explain your refactoring through comments in the original files.).

Good Coding Practices (1 mark)

- Variables and function names must be named appropriately.
- Looping and conditional statements must follow conventional and best-practice principles.
- Variable scope, span, and live-time minimization.

- All code duplication must be eliminated.
- The use of appropriate line spacing and indentation to improve code readability.
- Inclusion of comments to explain code blocks.
- Functions performing one task only.

Hand-in: Submit your **program (that includes ALL your files such as .sln)** via **stream** in a **zip** file.

If you have any questions or concerns about this assignment, please ask the lecturer well in advance. If you have questions regarding marking, then please contact Tong Liu directly.