

Assignment 4

Deadline:	Hand in by midnight Friday, 16th January 2015 and demonstrate at the following Lab session (19 th January 2015). The assignment is to be submitted via Stream.
Evaluation:	8 marks – which is 8% of your final grade.
Late Submission:	<i>1 mark deduction per day late</i>
Marking and Penalty	The assignment will be marked face-to-face with a tutor <i>Failure to attend the face-to-face marking session with your tutor (without prior alternative arrangement) will result in your marks being reduced by 50% (out of the possible maximum of 8 marks).</i>
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 acknowledge your source.
Purpose:	To learn to develop software using appropriate data structure, file input/output, and the TDD approach.




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 analysed, some statistics about the contents should be displayed.


The statistics should include the following metrics:

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		

For each metric, the word that fulfil the metric should be displayed. For example, if the word ‘supercalifragilisticexpialidocious’ is the longest word, then the metric for the ‘shortest word’ should display as follows:

“The longest word: ‘supercalifragilisticexpialidocious’ (34 characters).”

If there are two or more words that fulfil a particular metric, then all of the words that should be displayed in the statistics output. For example, if the words ‘the’ and ‘a’ occur 10 times in a file, and all other words in the file occur at most 9 times, then the words ‘the’ and ‘a’ fulfil the ‘most common word’ metric, and both of the words should be printed out in the display.

 The application should also include functionality to search for a specific word or words of a certain length. 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, all words of that length should be displayed (in sorted order).

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).

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 100% code coverage for user-defined methods (this excludes event-handlers).

Marking Criteria:Functionality (4%)

- The display of appropriate statistics (2% - 0.5% for each metric)
 - o deductions may be made for display of metrics that do not fulfil the requirements.
- Search functionality (1%)
 - o Search by word and appropriate display of search result (0.5%)

- Search by word length a.
- 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%)

The data structure (e.g. collections and arrays) that has been taught in the lectures so far should be used appropriately.

Proper Error Handling Mechanisms (1%)

The error handling mechanisms that have been taught in the close, including if/else statements, throw statements, and try/catch/finally statements must be used appropriately

Use of TDD (1%)

The application should be developed using TDD approach. Unit tests should be written for user-defined methods. Comprehensive unit tests is expected.

Good Coding Practices (1%)

Your application should be developed based on good coding practices, such as (not exhaustive):

- Appropriate naming of variables and function names
- Appropriate handling of loops and conditional statements
- Code modularity - the application must be broken down into subroutines/functions that perform one task each.
- Elimination of code duplications
- Appropriate comments
- Tidy presentation of code (use of appropriate line spacing, indenting, etc)\
- The application of code refactoring when necessary

Hand-in

Submit your **program** via **stream** in a **zip** file. Please contact the paper coordinator if you have any questions about the assignment (s.suriadi@massey.ac.nz). Make sure you zip your **entire** solution folder so that the .sln as well as the relevant project file are included.

If you have any questions or concerns about this assignment, please ask the lecturer well in advance.