UNIVERSITY OF Hull

DEPARTMENT OF COMPUTER SCIENCE
ASSESSMENT DESCRIPTION 2016/17
(EXAM TESTS WORTH ≤15% AND COURSEWORK)

MODULE DETAILS:

Module Number:	08230	Trimester:	2
Module Title:	Software Engineering		
Lecturer:	Dr L Bottaci		

COURSEWORK DETAILS:

Assessment Number:	1	of	1		
Title of Assessment:	Page Composition				
Format:	Program	Report			
Method of Working:	Individual				
Workload Guidance:	Typically, you should expect to spend between	40	and	80	hours on this assessment
Length of Submission:	This assessment should be **no** more than: *(over length submissions* ***will be*** *penalised as per University policy)*	Code submission, NA Report of 500 words. Words counted wherever they appear, in text, code, diagrams, images, etc.			

PUBLICATION:

Date of issue:	22 February, Wk 26 (4)

SUBMISSION:

ONE copy of this assessment should be handed in via:	Canvas		If Other (state method)	
Time and date for submission:	**Time**	2pm	**Date**	Thurs 27 April Wk 35 (10)
If **multiple hand–ins** please provide details*:*				
Will submission be scanned via TurnitinUK?	No	Submit single zip file containing VS solution and report. Solution top level folder must contain report as pdf file. Students are reminded they can submit **ONLY ONE** file and must ensure they upload the correct file.		

The assessment must be submitted **no later** than the time and date shown above, unless an extension has been authorised on a *Request for an Extension for an Assessment* form which is available from: http://www2.hull.ac.uk/student/registryservices/currentstudents/usefulforms.aspx

If submission is via TurnitinUK within Canvas staff must set resubmission as standard, allowing students to resubmit their work, though only the last assessment submitted will be marked and if submitted after the coursework deadline late penalties will be applied.

MARKING:

Marking will be by:	Student Number

ASSESSMENT:

The assessment is marked out of:	100	and is worth	100	% of the module marks
N.B If multiple hand-ins please indicate the marks and % apportioned to each stage above (i.e. Stage 1 – 50, Stage 2 – 50). It is these marks that will be presented to the exam board.				

ASSESSMENT STRATEGY AND LEARNING OUTCOMES:
The overall assessment strategy is designed to evaluate the student's achievement of the module learning outcomes, and is subdivided as follows:

LO	Learning Outcome	Method of Assessment *{e.g. report, demo}*
1	Show evidence of comprehension of object oriented and component based software engineering.	program output, report
2	Discern design patterns in systems, critically analysing and producing designs based on these patterns.	program output, report
3	Design a system by exploiting patterns and relevant object-oriented software engineering methods and explain/justify the approaches used.	program output, report

Assessment Criteria	Contributes to Learning Outcome	Mark
Report content.	1, 2, 3	20
Extent to which submitted program executable fulfills requirements.	1, 2, 3	80
Program does not conform to submission format.		Lose up to 10 marks
If program fails to execute then program assessed by inspection and program component mark capped at 50%.		

FEEDBACK

Feedback will be given via:	Assessment of program performance	Feedback will be given via:	Report comments
Exemption (staff to explain why)			
Feedback will be provided no later than 4 'teaching weeks' after the submission date.			

This assessment is set in the context of the learning outcomes for the module and does not by itself constitute a definitive specification of the assessment. If you are in any doubt as to the relationship between what you have been asked to do and the module content you should take this matter up with the member of staff who set the assessment as soon as possible.

You are advised to read the **NOTES** regarding late penalties, over-length assignments, unfair means and quality assurance in your student handbook, which is available on Canvas - https://canvas.hull.ac.uk/courses/17835/files/folder/Student-Handbooks-and-Guides.

In particular, please be aware that:

- Your work has a 10% penalty applied if submitted up to 24 hours late
- Your work has a 10% penalty applied and is capped to 40 (50 for level 7 modules) if submitted more than 24 hours late and up to and including 7 days after the deadline
- Your work will be awarded zero if submitted more than 7 days after the published deadline.
- The overlength penalty applies to your written report (which includes bullet points, and lists of text you have disguised as a table. It does not include contents page, graphs, data tables and appendices). Your mark will be awarded zero if you exceed the word count by more than 10%.

Please be reminded that you are responsible for reading the University Code of Practice on the use of Unfair means (http://www2.hull.ac.uk/student/studenthandbook/academic/unfairmeans.aspx) and must understand that unfair means is defined as any conduct by a candidate which may gain an illegitimate advantage or benefit for him/herself or another which may create a disadvantage or loss for another. You must therefore be certain that the work you are submitting contains no section copied in whole or in part from any other source unless where explicitly acknowledged by means of proper citation. In addition, **please note** that if one student gives their solution to another student who submits it as their own work, **BOTH** students are breaking the unfair means regulations, and will be investigated.

In case of any subsequent dispute, query, or appeal regarding your coursework, you are reminded that it is your responsibility, not the Department's, to produce the assignment in question.

Assignment Details

Requirements

Write a program to compose a page of text given a sequence of alphanumeric strings as input. Some of the strings may be words. A word consists of a non-empty sequence of lower case letters from a-z. A word must contain a vowel and words of length greater than 3 must contain at least two. All the vowels of a word must appear in alphabetical order. A line is a sequence of characters. The input words only must be placed on lines. A sequence of lines constitutes a page. All lines should end with the line termination character (i.e. '\n'). The length of a line is the sum of the characters on that line excluding the final '\n'. The various text formats required are listed below:

1. Fill – The position of any character on a line defines a column. The first word on any line must begin at the first column. The words should be concatenated, preserving the input order, together with a single space separating any two words on the same line. No other spaces should be present on the line. A wrap column is a non-negative integer used to limit the length of a line. No line containing two or more words may be longer than the wrap column. Only when no additional words may be placed on a line should the additional words should be placed on the next line.

2. FillSoft – The page should be formatted as in Fill, above, but in addition to the wrap column, there is a soft wrap column that is less than or equal to the wrap column. A

line containing two or more words may not have a length greater than the soft wrap column unless the position of the line on the page is less than or equal to half the number of lines on the page.

3. FillAdjust - the words should be formatted as in Fill, above, except that for lines with two or more words, one or more spaces must separate any two words on the same line and the last letter of the last word on any line must be on the wrap column. The number of spaces between words should, as far as possible, be equal. Where the inter-word space counts are not equal, as far as possible, the larger space counts should separate the words with the highest total number of vowels.

4. LineMoment - The moment of a letter in a word about a column is the product of the distance of a letter from the column and the value of the letter. The distance of a letter from a given column is the column of the letter less the given column. The value of a letter is the position of the letter in the alphabet, i.e. a = 1, b = 2, etc. The moment of a word is the sum of the moments of the letters in the word. The moment of a line about a given column is the sum of the moments of the words on that line. The page should be formatted as in Fill, above except that words may be separated by one or more spaces and the absolute value of the moment of each line about a given column should be minimal. and the most separated words should be at the front of the line

5. FillSet – The page should be formatted as in Fill, above except that the order of the words on the page need not correspond to the order of the words in the input. The number of lines on the page should be minimal.

The required program may assume the following pre-condition. The input sequence of strings resides in an Xml file called "input.xml" (all input must come from this file only). The input file also contains the required format specification as a string, e.g. "Fill", "FillSoft", etc. The input file will also contain any other column parameters of the format. Sample Xml files will specify the file syntax. The page produced by the program should be written to the file "page.txt" (no other file must be written) as a sequence of lines, each ending in '\n'. The input file is located in the program working directory. The output file must be placed in the same directory.

The code provided is useful but there is no guarrantee that it is free from faults. The Visual Studio solution also contains a sample input file in the bin\Debug directory of the main project directory.

The program output file will be checked mechanically by executing the program on an input file. It is essential that all output must be to the file only and be in the prescribed format. There should not be any output whatsoever to the Console.

The program is not expected to execute very large inputs and any program execution that continues for over one half second will be terminated and any output discarded. This should affect only the extremely inefficient implementations. The program should not throw any unhandled exceptions.

Finally

Note that, like real world requirements, the above requirements can be expected to be reasonably accurate, reasonably complete and reasonably consistent but minor problems may be present. Solving such problems is part of the assessment. Again, like real world requirements, clarification or minor modifications to the above requirements may be issued, via University email, after the publication of this document.

You should submit a Visual Studio solution, with a directory structure equal to the supplied VS solution, i.e. no directories should be added, removed or renamed. Files may be added to the existing folders. The name and location of the executable program (bin\Debug directory of the main project directory) should not be changed. The VS solution version should be the same as the version of the supplied VS solution.

Before submission, build the program and check that it executes correctly. You must use the supplied submission conformance program to check and compress (zip) your VS solution. If the program passes the conformance checks the Visual Studio solution will be compressed into a zip file suitable for submission to Canvas. Do not compress the program by calling a zip program directly.

Report

Write a concise report giving an abstract description of your program. Explain the architecture and design of the program and point out the use of any object oriented design patterns. Very short program fragments may be included to illustrate the description..

The report, a pdf document file called “report.pdf”, should reside in the top level directory of your Visual Studio solution. The report should not contain more than 500 words. A word is considered to be anything that is to be read and thus includes symbols in code. The number of words in a fragment of code is the number of words reported by Word. A word is counted irrespective of where it occurs, i.e. in text, a table, drawing, image etc.

Assessment

This is a software engineering task and not simply a programming task. As such, an important part of the task is to analyse and understand the requirements. Working with existing code and frameworks is also a key feature of software engineering as is design, and testing.

The program submitted will be assessed by execution on a set of test inputs. The program output file will be checked mechanically and in detail and marks will be awarded for each correct test execution. The code will, however, be inspected if it does not compile or fails an excessive number of tests. The mark for such a program is capped at 50%.