Assignment

CSP2104 Object-oriented Programming with C++

Overview

Due date:

Part 1 (10 Marks): **Monday, 23rd April 2018 (5PM)**

Part 2 (30 Marks): **Friday, 25th May 2018 (5PM)**

This assignment is to create an application which loads an English dictionary and then performs certain tasks using that dictionary. The assignment is worth 40 marks (or 40% of the total mark for the unit). The assignment contains tasks of varying complexity and students are expected to consult sources outside of the unit materials in order to accomplish them.

Related Learning Outcomes from Unit Outline

On completion of this assignment students should be able to:

- Formulate computer algorithms using the operations, control structures and classes provided in the C++ programming languages;
- Write, test and debug computer programs written in C++;
- Design and implement a class library as an abstraction, using the facilities of the C++ language and environment;
- Implement efficient exception handling mechanisms in user-defined classes;

General

Whilst you are allowed to discuss aspects of the assignment with your colleagues, all material submitted for marking must be your own work. Plagiarism and other forms of academic misconduct will be dealt with as per the relevant ECU policy.

Your assignments will be processed with plagiarism detection software.

Submission Instructions

The assignment submission should be made via the Unit's Blackboard assignments site. Penalties will be applied for late submission.

Your submission must be a single zipped file containing your entire visual studio project. The file name should be in the format of <yoursurname_studentnumber.zip>. For example, smith_1001234.zip.

Your submission must be uploaded to BlackBoard before or on the due date unless you have an approved extension. Extensions must be applied for in writing before the due date as per University policy.

Also ensure that you keep a backup copy of all documentation and program files.
Submit your work as a visual studio (2017) project. Make sure it compiles.

Part 1 (10 Marks)

Write a program that when executed will parse the dictionary file, **dictionary2018.txt**, provided with this assignment. It will load each record from the file into a new instance of a **Word** class and add this instance to a list of Word objects contained in an instance of a **Dictionary** class. Your software will then display a menu of tasks it can perform. It will then prompt the user to enter a number corresponding to one of the following menu items:

Basic Tasks

1. Prompt the user to enter a word. If that exact word is in the dictionary, print the word's definition, pre-pended with the word type using the following scheme: Noun (n.), Verb (v.), Adverb (adv.), Adjective(adj.), Preposition(prep) MiscWords(misc.),ProperNouns (pn.), NounAndVerb (n. v.). If the word is not in the dictionary print 'word not found.'
2. Find the longest word in the dictionary.
3. Find the word(s) which end in 'logy' and contain seven letters or less in total.

Classes to Create (for Part 1)

This is to help you get started with your dictionary. Words in italics indicate what you MUST call your classes, methods or fields if you wish to receive marks for your effort.

Create a *'Word'* class

- Fields:
  - *word* (string) The word in the dictionary
  - *type (use an 'enum' to store types)*
  - *definition* (string)
- Methods:
  - Getters for *word, definition and type*
  - *printDefinition – print the word's definition to the console in accordance with the requirement for basic task 1.*

Create a '*Dictionary'* class which

- maintains an appropriate STL container of Word objects
- loads the dictionary file into its array of Word objects
- Performs the tasks needed for this assignment.
- provides the methods:
  - *loadDictionary*() - (loads the dictionary file into its array of Word objects)
  - *Other appropriate methods so as to implement the tasks.*

Appendix 1 at the end of this document outlines the structure of the dictionary file.

Part 2 (30 Marks)

An extension of Part 1. Implement these additional tasks and add them to the menu of the program.

Basic Tasks, in addition to those from Part 1:

1. Prompt the user for a word, and report all words that rhyme (by checking if the last 3 letters are the same)

2. Print a word’s scrabble score following its definition. (this task does not need its own menu entry – can be implemented as part of the definition search from Part 1 of the assignment). The rules for calculating a word’s scrabble score are given in Appendix 2 at the end of this document.

Intermediate tasks

3. Prompt the user for a word, and report all words that are anagrams of the word (e.g. “admirer” and “married”)
4. Let the user enter a string of letters, return the word with the highest scrabble score which can be made using these letters.

Advanced tasks

5. Create a glossary generator - Create functionality that will:
    a. Parse the file “many_english_works.txt” one word at a time. Every time a word is found that is in the dictionary increment the words *usageFrequency.*
    b. Parse the text “new_work.txt” one word at a time, for every word that is also in the dictionary and isRareWord(), a method that you implement, returns true print the word and its definition. Careful not to print the same word twice!
    c. Save the glossary to a text file.

Marking Guide

Earn marks

Part 1	**Marks**	**Note**
Word class and Dictionary class correctly implemented.	2	
Dictionary file loaded and parsed correctly.	2	
3 Basic Tasks	6	
Part 2	**Marks**	**Note**
2 basic tasks	10	
2 intermediate tasks	10	
1 advanced task	10	Create classes and methods appropriate to the task.

Lose marks

Problem	**Marks (up to)**	**Note**
Poor programming practice.	5	Lack of commenting Magic numbers instead of constants Poor variable and function names
Input validation fails	5	I will try to break your program with dodgy input.
Program crashes	5	If your program crashes during execution then you will lose marks.

Appendix 1: Format of the dictionary file:

Notes about dictionary.txt

- 106,184 word definitions
- Text format (ascii)
- 3 lines per definition
    - The word [Type]
    - The definition (on one line)
    - Blank line
- Word
    - Only uses characters A-Z a-z and the hyphen '-'
    - No punctuation or similar
    - Abbreviations are given without the '.' For example, "e.g." would be "eg"
    - No words are presented with spaces, the words are joined OR a hyphen is used. e.g. "bumble bee" is "bumblebee"
    - The language conversions are made eg: é=e æ=ae ö=o
    - No word has more than 45 characters
    - ALL words are in lower case, even proper nouns.
- Type, a single word (see table 1) enclosed in square brackets, appearing on the same line as the word, separated by a space from it;
- Definition
    - Multiple definitions are separated by a semicolon;
    - No Definition has more than 6014 characters
    - The language conversions are made eg: é=e æ=ae ö=o

• The definitions were not written by your lecturer or ECU. We take no responsibility for any inaccuracies or the content.

- The definitions are from the GCIDE project, made available under the terms of the GNU general Public License, GCIDE_XML is necessarily also published under those terms. See the file gpl.txt or <http://www.gnu.org/copyleft/gpl.txt>.

Table 1 Types present in the dictionary and the key Words used to denote them.

Word	Count	Meaning
v	9715	verb ("run", "jump")
n	59652	noun ("cat", dog")
adv	3413	adverb ("slowly")
adj	28453	adjective ("big", "glowing", "inexpensive")
prep	96	preposition ("beneath", "against")
pn	874	proper noun ("Perth", "Edith Cowan")
n_and_v	156	This word is a noun and a verb ("Rain", "Phone")
misc	3899	other words e.g. "shh", "and", but", "arg", "more" and prefixes

Appendix 2: Calculating Scrabble Score (should be a method in the Word class)

Score the word as per the table below. Return an integer. Return 0 if the word is not allowed.

Letter	Score	Letter	Score	Letter	Score	Letter	Score
A	1	I	1	Q	10	Y	4
B	3	J	8	R	1	Z	10
C	3	K	5	S	1		
D	2	L	1	T	1		
E	1	M	3	U	1		
F	4	N	1	V	4		
G	2	O	1	W	4		
H	4	P	3	X	8		

Any word with a hyphen is not allowed and returns a zero score. *MiscWords* and *ProperNouns* are not allowed in scrabble. Only the following classes may return a non-zero scrabble score when *calculateScrabbleScore()* is called:

- Noun
- Verb
- Adverb
- Adjective
- *Preposition*
- *NounAndVerb*