CO1107 Algorithm, Data Structure and Advance Programming
CW1 – Group Assignment

Assessment Number	1
Contribution to Overall Mark	40%
Submission Deadline	16/02/2023
Member per Group	3 Members (Max)

This assessment aims at testing your ability to develop a robust Python code which solves a given problem. This assessment looks at developing a Python program using various programming constructs and data types such as strings, lists, set and dictionary as well as function and file handling. All the required functionalities are specified and you should implement each of them with appropriate exception handling.

PART I: (70 Marks)

This assignment is about a single player scrabble game. Assume that once the program runs, it randomly prints the 7 tiles and their scores. Then the user is asked to enter a word. The program then should check if the word entered by the user is valid or not. A valid word must satisfy all the following criteria:

1. The word must consist of only English letters. For example, the word "He3lo" is invalid because it contains a number. Similarly, the word should not contain whitespace, for example, "Hello World" is invalid.
2. If the above rule is met, your program must ensure that the word exists in the dictionary '*dictionary.txt'*. Note that your program should convert the word into upper case. Note that the dictionary does not contain all English words.
3. Then your program must check whether the word can be made using the 7 tiles generated initially. For example, if the tiles are "A H C E L M O", then the word "HELLO" is invalid because there is only one L in tiles, but the word HELLO contains two letters "L".

Once the user has entered a valid word, the system must compute and print the score of the given word. **Hint**: The score of a word is the total score of all the letters in the word where score of each letter is given in *scores.txt.*

In order to complete this assignment, you will be provided with a few files:

- **dictionary.txt** which contains a list of words found in an online dictionary.
- **score.txt** which contains score for each English letter.
- **tiles.txt** which contains all valid English letters.

Given the above scenario, write a program for each one of the following tasks and save them into a separate python file; for example: PartI_Task1.py, PartI_Task2.py, etc.

Task 1:
Read the scores, tiles, and words from **scores.txt**, **tiles.txt** and **dictionary.txt** respectively and insert them into a relevant list (for example Scores, Tiles, Dictionary)

Hint: You may consider using other data types instead of a list to store the data if you think it would be more appropriate.

[10 Marks]

Task 2:
Write a function called ***onlyEnglishLetters()*** that accepts a word as parameter and returns True if it contains only English letters, and False otherwise. You must write the function to work for any input data. The function should return False if the input data is invalid.

Hint: A word is not valid if it contains numbers or spaces. For example, the output for the word ***HELLO***, ***HE LLO*** and ***HE3LLO*** must be **True**, **False** and **False**, respectively.

[10 Marks]

Task 3:
Write a function called ***getLetterScore()*** that accepts a letter and returns its corresponding score. You must write the function to work for any input data. The function should return 0 if the input data is invalid.

Hint: In order to test your function, you should reuse your code from Task 1 to read *scores.txt* and insert them into the list ***Scores.***

[10 Marks]

Task 4:
Write a function called ***getWordScore()*** that accepts a word and returns the score of that word. You must reuse ***getLetterScore()*** written in Task 3 in this function. You must write the function to work for any input data. The function should return 0 if the input data is invalid.

[10 Marks]

Task 5:
Write a function ***canBeMade()*** that accepts two parameters: word & myTiles. Then check if that word can be made with the given Tiles. You must write the function to work for any input data. The function should return False if the input data is invalid.

For example, if myTiles = ['T','Y','S','E','U','W','I'] then the word “SWET” can be made using myTiles.

But if myTiles = ['T','Y','S','E','U','W','I'] then the word “SWEET” can not be made using myTiles.

[15 Marks]

Task 6:
Write a function called ***isValid()*** that accepts three parameters (word, myTiles, dictionary) and return True if the word meets all three rules listed in the program specification. You must write the function to work for any input data. The function should return False if the input data is invalid.

Hint: You should call some of the functions that you have done in previous tasks.

[15 Marks]

Task 7: (Optional)

Write a function that find a valid word with the highest score and display this to the user along with its score. [Not assessed]

PART II: (30 Marks)

Write a complete program to simulate a single player scrabble game; once the program runs, it randomly prints the 7 tiles and their scores. Then the user is asked to enter a word. The program should check if the word entered by the user is valid or not based on the criteria listed in Part I. Note that the user input is NOT case-sensitive; for example, HELLO and HelLO are to be considered the same. The user should repeatedly be asked to enter a word until they enter a valid word, or enter "&&&" indicating that the user has given up and does not want to enter another word.

If the user enters a valid word, you should print "You got it right, this is a valid word".

If the user enters an invalid word, you should print the ***first failed condition*** from the 3 criteria described in part I. One of: "Only use English letters…", "There is no such word in the dictionary" or "This word cannot be made from these tiles".

If the user enters &&& you should print "Thanks for using this application, better luck next time!!!".

Once the user has entered a valid word, the system must compute and print the score of the given word. **Hint**: The score of a word is the total score of all the letters in the word where score of each letter is given in *scores.txt.*

Important Note: Your program **must contain** all the functions from Part I.

Below are some sample outputs:

Sample Output 1

Generating Random Tiles…
Tiles: T E E E E D H
Scores: 2 1 1 1 1 3 3
Enter a word: TED
You got it right, this is a valid word
Score of this word is: 6

Sample Output 2

Generating Random Tiles…

Tiles: O L I T E G O

Scores: 2 3 1 2 1 4 2

Enter a word: TIL3E

Only use English letters…

Enter a word: TILE

You got it right, this is a valid word

Score of this word is: 7

Sample Output 3

Generating Random Tiles…

Tiles: Q N R L P V I

Scores: 20 3 3 3 5 10 1

Enter a word: PVI

There is no such word in the dictionary

Enter a word: RLP

There is no such word in the dictionary

Enter a word: &&&

Thanks for using this application, better luck next time!!!

How to submit

For this assignment, you need to submit the following:

1. A signed coursework cover – this should include the names of all the students involved in the work submitted.
2. The source code of your implementation (either the .py or .ipynb file) for all the **questions you have answered separately**. You should include comments in your code stating the aim of the program; who is the author; and for each function you have written, what are the pre-conditions and post-conditions.

Please put all the 8 files including the optional task 7 (.py or .ipynb file), Tasks 1-7 and the full program, together with signed declaration into a zip file CW1_YourGroupNum.zip (e.g. CW1_Group1.zip) and then submit your assignment through the module's blackboard site by the deadline. Note that to submit, you need to click on the Coursework link on Blackboard and then upload your zipped file.

Important Notes:

1. All members must contribute. **All members will be called for an interview after the submission to demonstrate the understanding of their code**.
2. Please **DO NOT** share your solutions with others; all parties involved in plagiarism and collusion will get **zero mark** for the assignment.
3. You must test your code against a number of test cases; you may still get some marks even if your implementations have bugs, depending on how close your code is to the correct algorithm.
4. This assignment has a total of 100 marks and contributes to 40% of your total mark.