1801ICT Individual Assignment – Milestone 2
Due end of Week 11 (2nd October by 11:59pm) (100 marks)

This assignment requires you to solve problems and write software independently, and to integrate various aspects of C++ programming that you have learned this trimester. You must write ALL the C++ program code. Lecturer and tutors are happy to answer questions on this assignment.

Code, internally document and test each function. Demonstrate the workings of your program.

Requirements:

- You are responsible for designing programs which addresses the requirements of the assignment.
- You should design your algorithms and your overall programs so that your code is efficient and clean.
- You should test the components of your programs separately, but you should end up with working programs at the end.
- **The version of C++ used for marking will be C++ 14.**
- **You MUST submit one zip file containing three program source files (word_guess.cpp, word_sum.cpp, word_seq.cpp) ONLY. The zip file MUST be named <Your last name>-<snumber>-A2.zip. So student Joe Bloggs, s1234567 would submit the following file:**

bloggs-s1234567-A2.zip

Within the zip file, your C++ source files are your updated versions of the supplied files word_guess.cpp, word_sum.cpp (do not change the names of these files) and the file word_seq.cpp.

SUBMISSION [TO BE SUBMITTED END OF WEEK 11 VIA L@G] (2nd October by 11:59pm)

ASSIGNMENT SPECIFICATIONS

All programs should be submitted online by the due date and time. If you need any help, please ask as we can give you hints and suggestions.

Part 1: Word Guessing (start with the supplied word_guess.cpp C++ file) – sample solution has 65 semi-colons

In this part of the assignment, you will build a program that bends the rules to trounce its human opponent time and time again. In case you aren't familiar with the Word Guessing game, the rules are as follows:

1. One player chooses a secret word, then writes out a number of dashes equal to the word length.
2. The other player begins guessing letters. Whenever they guess a letter contained in the hidden word, the first player reveals each instance of that letter in the word. Otherwise, the guess is wrong.
3. The game ends either when all the letters in the word have been revealed or when the guesser has run out of guesses.

Fundamental to the game is the fact the first player accurately represents the word they have chosen. That way, when the other players guess letters, they can reveal whether that letter is in the word. But what happens if the player doesn't do this? Suppose that you are playing Word Guessing and it's your turn to choose a word, which we'll assume is of length four. Rather than committing to a secret word, you instead compile a list of every four-letter word in the English language. For simplicity, let's assume that English only has a few four-letter words, all of which are reprinted here:

ALLY BETA COOL DEAL ELSE FLEW GOOD HOPE IBEX

Now, suppose that your opponent guesses the letter 'E.' You now need to tell your opponent which letters in the word you've "picked" are E's. Of course, you haven't picked a word, and so you have multiple options about where you reveal the E's. Here's the above word list, with E's highlighted in each word:

ALLY BETA COOL DEAL ELSE FLEW GOOD HOPE IBEX

Every word in the word list falls into one of five “word categories:”

- ----, containing the word ALLY, COOL, and GOOD.
- -E--, containing BETA and DEAL.
- --E-, containing FLEW and IBEX.
- E--E, containing ELSE.
- ---E, containing HOPE.

Since the letters you reveal have to correspond to some word in the word list, you can choose to reveal any one of the above five categories. There are many ways to pick which category to reveal but we will steer your opponent toward a larger category in the hopes of keeping your options open. Adopting the larger category approach of choosing the largest of the remaining word categories you would pick the category ----. This reduces your word list down to ALLY COOL GOOD and since you didn't reveal any letters, you would tell your opponent that their guess was wrong.

Your program should do the following:

1. Read the file dictionary.txt, which contains the full contents of the Official Scrabble Player's Dictionary, Second Edition. This word list has over 120,000 words.

2. The user will enter the word length and the maximum number of allowed guesses on the command line.

3. Play the Word Guessing game as described below:

    a. Construct a list of all words in the English language whose length matches the input length.
    b. Print out how many guesses the user has remaining, along with any letters the player has guessed, the current blanked-out version of the word and the number of words remaining (the program would not normally display the number of words remaining but it is useful for testing).
    c. Prompt the user for a single letter guess, re-prompting until the user enters a letter that she hasn't guessed yet. Make sure that the input is exactly one character long and that it's a letter of the alphabet.
    d. Partition the words in the dictionary into groups by word category.
    e. Choose a word category and remove all words from the word list that aren't in that category and report the position of the letters (if any) to the user. Subtract a remaining guess from the user.
    f. If the player has run out of guesses, pick a word from the word list and display it as the word that the computer initially “chose.”
    g. If the player correctly guesses the word, congratulate them and exit.

    Required Program Output Format:

    ```
    ./a.out 6 6
    Enter a letter: a
    ______, Guesses left = 5, Words left = 8154
    Enter a letter: i
    ______, Guesses left = 4, Words left = 4621
    Enter a letter: e
    ____e_, Guesses left = 3, Words left = 1311
    Enter a letter: u
    ____e_, Guesses left = 2, Words left = 741
    Enter a letter: b
    ____e_, Guesses left = 1, Words left = 644
    Enter a letter: d
    ____e_, Guesses left = 0, Words left = 373
    Sorry, the word was choker
    ```

Part 2: Word Sums (start with the supplied word_sum.cpp C++ file) – sample solution has 74 semi-colons

A Word Sum is a puzzle in which the digits in a correct mathematical expression, such as a sum, are replaced by letters and where the puzzle's words are in the dictionary. For example, if D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9 and Y = 2 then the following is true:

```
 SEND        =>  9567
 MORE        =>  1085
 ---------       -------
MONEY         => 10652
```

Two conditions are assumed: firstly, the correspondence between letters and digits is one-to-one and different letters represent different digits. Secondly, the digit zero does not appear as the left-most digit in any of the numbers.

Write a program that, given a Word Sum, determines the correct assignment of digits to letters so that the numeric summation is correct. Demonstrate your program with the following problems:

SEND + MORE = MONEY
EARTH + AIR + FIRE + WATER = NATURE
SATURN + URANUS + NEPTUNE + PLUTO = PLANETS

Required Program Output Format:

Problem: SEND + MORE = MONEY, CPU = 0.181375
D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9, Y = 2, Attempts = 1110106

Problem: EARTH + AIR + FIRE + WATER = NATURE, CPU = 0.398003
A = 7, E = 6, F = 8, H = 2, I = 0, N = 1, R = 4, T = 3, U = 5, W = 9, Attempts = 1354807

Problem: SATURN + URANUS + NEPTUNE + PLUTO = PLANETS, CPU = 0.270416
A = 2, E = 9, L = 6, N = 3, O = 8, P = 4, R = 0, S = 1, T = 7, U = 5, Attempts = 760286

Part 3: Word Sub-sequences (create file word_seq.cpp) – sample solution has 40 semi-colons

Read the file *dictionary.txt* and list the top 10 occurrences of sub-sequences of 2, 3, 4, and 5 letters. Your program must only read the file once and the output should be:

Found 584 sub-sequences of length 2
on 12027
at 12534
te 13163
ed 13182
re 13581
ng 13880
ti 14640
er 21784
es 23185
in 23567

Found 6686 sub-sequences of length 3
nes 3603
ate 3728
ter 3927
tio 3989
ies 4031
ess 4267
ati 4689

ion 4826
ers 5428
ing 12268

Found 36124 sub-sequences of length 4
ally 1152
ment 1184
ring 1245
able 1322
ling 1479
ions 1851
ting 2365
ness 2579
atio 2654
tion 3855

Found 85927 sub-sequences of length 5
bilit 490
inter 504
ering 557
icall 601
cally 617
ities 628
eness 674
ating 813
tions 1554
ation 2627

CPU time = 0.257334 seconds

Note: You must actually write **all** C++ code submitted.

Marking

The assignment will be marked out of 100 and marks will be allocated as follows:

1. Quality of C++ code (e.g. indentation, appropriate naming conventions, appropriate use of C++ data structures and algorithms, method structure, efficiency,…): **40 marks**
2. Implementation of Part 1 : **20 marks**
3. Implementation of Part 2 : **20 marks**
4. Implementation of Part 3 : **20 marks**