# 2807ICT Assignment – Semester 1, 2017

This assignment can be done either individually or as a team of at most two students. The submission date is as specified in the Course Profile and the submission method will be communicated during semester.

## Word Guessing

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 B**E**TA COOL D**E**AL **E**LS**E** FL**E**W GOOD HOP**E** IB**E**X

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 – perhaps you want to steer your opponent toward a smaller category with more obscure words, or toward a larger category in the hopes of keeping your options open. If we adopt the larger category approach and always choose 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, which should be more than enough for our purposes.
1. Prompt the user for a word length, re-prompting as necessary until they enter a number such that there's at least one word that's exactly that long. That is, if the user wants to play with words of length -42 or 137, since no English words are that long, you should re-prompt them.
2. Prompt the user for a number of guesses, which must be an integer greater than zero. Don't worry about unusually large numbers of guesses – after all, having more than 26 guesses is clearly not going to help your opponent!
3. Prompt the user for whether they want to have a running total of the number of words remaining in the word list. This completely ruins the illusion of a fair game that you'll be cultivating, but it's quite useful for testing
4. Play the Word Guessing game as described below:
    1. Construct a list of all words in the English language whose length matches the input length.
    2. Print out how many guesses the user has remaining, along with any letters the player has guessed and the current blanked-out version of the word. If the user chose earlier to see the number of words remaining, print that out too.
    3. 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.
    4. Partition the words in the dictionary into groups by word category.
    5. 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. If the word category doesn't contain any copies of the letter, subtract a remaining guess from the user.
    6. 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."
    7. If the player correctly guesses the word, congratulate them.
    8. Ask if the user wants to play again and loop accordingly.

**Notes:**

1. You must actually write **all** Python code submitted in the PyLab environment.

2. You must produce Word documents describing your code and the testing you have performed. A tool for generating UML diagrams can be found at http://horstmann.com/violet/

**Marking**

This assignment is worth 19% of your final grade. The assignment will be marked out of 100 and marks will be allocated as follows:

1. Documentation of software and testing : **15 marks**
2. Quality of Python code (e.g. appropriate naming conventions, class structure, method structure,...) : **15 marks**
3. Implementation including testing : **70 marks**

Please note that all submitted assignments will be analysed by a plagiarism detector that is specifically designed for assignment submissions containing Python source code.