**COSC 1336 Exam 3 Lab (open book, open note)**

Python lab for Exam 3 (85 points).

* Write a spell checker program (85 points)
* Provide a word count status message (extra credit, 5 points)

The program is described in three parts. You create one new Python program from scratch. Details for each part follow:

**Part 1**. (35 points)

1. Start IDLE and open a new file with the name **MW19\_X3\_Lastname.py**.
2. Add a comment at the top with your name, course (COSC1336) and brief description. This is a spell-checker program.
3. Write a Python program as follows (**important:- follow the specification**):  
   1. **Define a function: notice()** that returns a message which briefly describes what the program does. The message returned will be:  
       **Spell checking program by <your name> for Exam 3 lab**

**IPO:**   
 **Inputs: parameter1:**  exam\_number (when called, exam\_number will be 3),  
 **parameter2:**  name (when called, name will be your name),  
 **Processing:** Create message of type string. Use the exam\_number and name.   
  
 Example message returned by: notice(3, ‘Mary Lew’):  
 ***Spell checking program by Mary Lew for Exam 3 lab*** where:  
 exam\_number is 3 and name is Mary Lew  
 **Output (return):** message (a string) as described above; no side-effects.

* 1. **Define a function: load\_dictionary(wordlist).**Here is the IPO:
     1. **Input:** list to put words into, named **wordlist**
     2. **Processing:**
        1. Open a file called **dictionary.txt**. (Don’t ask this for name. Always use the name dictionary.txt, which is provided.)
        2. If the file is not found, catch an exception, print a good error message (explain the problem) and return **False**; otherwise:
        3. In a loop, for each word in the file:
           1. Read a word from dictionary.txt.
           2. From the word, strip off the newline character. (Hint: rstrip)
           3. After stripping the newline character, put the word into **wordlist**, which was passed in as a parameter.
        4. Display the number of words you found in the dictionary.
        5. Close the file.
        6. return **True** (wordlist will contain many words)
     3. **Output: True** if wordlist was loaded with words, OR **False** if the file: dictionary.txt was not found  
         side-effects: display error message if file not found  
         display number of words in wordlist
  2. **Define a function main()** that does this:
     1. call notice() with arguments 3, <your name>; print the return string.
     2. create an empty list called **dictionary**
     3. call load\_dictionary(dictionary) # loads words into list dictionary
     4. print Good bye!
  3. call main()
  4. Test your program on the file: **dictionary.txt** (provided on Blackboard).
  5. At this point, your program should create and display the following output:  
     **Spell checking program by <your name> for Exam 3 lab  
     The dictionary has 80372 words.**
  6. Make sure your program does not crash if file **dictionary.txt** is not found.

**Part 2**.(35 points)

1. **Create a new function called spellcheck(dictionary).**
   1. **Input: parameter named dictionary, a list of words, previously filled up**
   2. **Processing:**
      1. **Create a loop that does this:**
         1. **Prompt: “Name of file to spell check (.txt optional, enter nothing to quit): “**
         2. if nothing is entered, quit gracefully (**say “Goodbye”**), return from the function; otherwise:
         3. if the filename does not contain a .txt at the end, add it.
         4. open the filename
         5. If the file to spell check is not found, catch an exception, print a good error message. Continue in the loop to try another name.
         6. **Loop until you get a valid filename or the user enters <nothing> to quit.** Here, <nothing> means only the Enter key is pressed. If user enters <nothing>, break out of the loop.
      2. Still inside the loop, after a valid file to spell check was opened successfully, do this...
      3. Provide a counter to keep track of which line number you are on. This is the line number of the file being spell checked.
      4. From your open a valid file, enter a new loop which does this:
         1. Read a line of text from the file being spell checked.
         2. Split the line of the spell check file into individual words. Hint: split()
         3. For each individual word, check to see if it is found in the list of words called dictionary.
         4. If the word is NOT found in the dictionary, output a message like:  
            **horss on line 5 not found**where **horss** is a word in the spell check file but not in the dictionary.
      5. Close the spell check file.
      6. Loop back up to spell check another file, until user enters nothing to quit.
   3. **Output: none;** side effects: displays misspelled words and line numbers.
   4. In main, call load\_dictionary(). Check the return value from load\_dictionary. If the return value indicates that load\_dictionary was successful, call spellcheck(); otherwise, stop. You cannot spellcheck a file if the dictionary did not load.
2. After Part 2, your program will also print a list of misspelled words (words in the spell check file but not in the dictionary). However, it does not correctly spellcheck the document. It marks some correctly spelled words as incorrect due to punctuation. To see the problems, test spell check on the file: **letter.txt** (provided).

**Part 3**. (5 points)

In this part, you debug your spellcheck to work better.

1. Enhance spellcheck(). In the enhanced version, do this:
   1. make sure that you are only checking valid words that contain alphabetic characters. That is, don’t look for words that contain whitespace or have extra punctuation. For example, words like: **Oh! my, goodness!** must be adjusted to **oh my goodness** before spell checking. To do this,
   2. from each word being checked for spelling, strip off whitespace characters.
   3. from each word being checked for spelling, remove extra punctuation. Remove pre and post: punctuation: period, comma, exclamation mark, quote, etc.  
      Do not remove punctuation inside the word. Do not change in-law to inlaw.
   4. Change uppercase letters in words from the file being checked into lowercase before checking to see if they are in the dictionary. (Otherwise, the word **Good** will be marked as misspelled, even though **good** is in the dictionary.)

**Part 4**. (5 points)

This last part is testing. Test on a missing “**junk**” or invalid “**::\/$!~**” file name. Program should display error message and loop back to prompt

Test on filename “**letter**” (provided). Your program should produce output like this (example):  
  
**Spell checking program by Mary Lew for Exam 3 lab**

**The dictionary has 80372 words.**

**Enter name of file to spell check (with or without .txt): letter**

**Selena on line 2 NOT found**

**horss on line 5 NOT found**

**ranned on line 6 NOT found**

**furr-way on line 6 NOT found**

**werds on line 7 NOT found**

**Plinkerton on line 11 NOT found**

**Good bye!**

Paste test output at bottom and submit the completed file on Blackboard.

**Extra Credit** (5 points) Keep track of the total number of words in the spell check file and the total number of misspelled words. **In addition to the spell check output above**, provide this status message:

**6 of 49 words not found in file: letter.txt**