COMP 1602 Computer Programming II

2017/2018 Semester 2

Assignment 3: Spell Checker for Different Languages

Date Due: *Sunday March 11, 2018*

Overview

This assignment requires you to write a program that finds spelling errors in a passage. The words are checked against a dictionary. Your program must also generate basic statistics such as the amount of different words in the passage and the most frequently occurring word.

Your program must cater for three languages: English (*E*), French (*F*), and Spanish (*S*). So, your program must create a dictionary for each language when requested to do so. The passages can either be in English, French or Spanish. The spelling of each word in a passage is checked using the appropriate dictionary.

Description

(a) Your program should display the following menu:

```
Spell Checker for Different Languages
---------------------------------------------

1. Load dictionary [English, French, Spanish]
2. Load passage
3. Display passage
4. Spell check passage (with statistics)
5. Quit

Please enter an option:
```

(b) When an option is selected, the appropriate action must be taken, after which the menu is re-displayed:

1. When this option is selected, your program should allow the user to choose a language:

```
Please enter E (English), F (French), S (Spanish) or m (Menu):
```

If the user chooses "m", control should return to the main menu and no change should be made to the dictionary. Otherwise, your program should load the appropriate dictionary in an array.

2. When this option is selected, the program should request the user to specify the name of the file containing the passage to be checked:

```
Please enter the name of file with passage or m (Menu): passage-en.txt
```

If the user chooses "m", control should return to the main menu. Otherwise, it should open the file and copy all the characters from the file to an array of characters terminated with the null character.

3. When this option is selected, the program should display the array of characters from Option 2. This is the set of characters from the passage which were read from the file. If the passage was not previously loaded via Option 2, an error message should be displayed.

4. When this option is selected, the program should spell check the passage using the dictionary that was specified in Option 1. If no dictionary was selected or the passage was not loaded, an error message should be displayed. Section (c) gives more information on the spell check process.

(c) (1) During the process of spell checking the passage, each word is extracted from the array of characters created in Option 2. A word consists of letters of the alphabet (both uppercase and lowercase), hyphens, or apostrophes. No other characters are permitted. It should be noted that in addition to words, the passage contains punctuation characters.

(2) The dictionary specified via Option 1 is searched for each word. If a word is not found in the dictionary, it should be stored in a separate array, containing only words with spelling errors.

(3) If a word is found in the dictionary, it should be stored once in an array of valid words. You should also keep track of the amount of times a valid word appears in the passage.

(4) After all the words are read from the passage, the spelling errors are displayed. As the spelling errors are being displayed, each one is checked to determine if it is due to a *single letter transposition* error. A single letter transposition error occurs when two **adjacent** characters of a valid word are mistakenly interchanged while typing, e.g.,

teh	the ("eh" with "he")
hoewver	however ("ew" with "we")
recieve	receive ("ie" with "ei")

If a single letter transposition error is detected, your program should display the corresponding valid word.

Please note that all the words in the dictionary are in lowercase letters.

(d) After completing the spell check process described in (c) above, your program should display the following information:

(1) The total number of words found in the passage (including words with errors).

(2) The total number of valid words found in the passage, as well as a list of each valid word and the number of times it appears in the passage.

(3) The amount of spelling errors detected in the passage.

(4) The amount of corrections which were suggested due to single letter transposition errors.

(5) The most frequently occurring word in the passage.

(6) If the number of words with spelling errors is more than half the total number of words in the passage, you should display a message like the following:

```
This passage does not seem to be in the language selected (French).
Please choose another language.
```

Files Required (Available at Course Web Site in myElearning)

File	Description
`dictionary-en.txt`	English dictionary
`dictionary-fr.txt`	French dictionary
`dictionary-es.txt`	Spanish dictionary
`passage-en.txt`	English passage
`passage-fr.txt`	French passage
`passage-es.txt`	Spanish passage

You can use other passages to test your program as long as the corresponding file is correctly specified in Option 2.

Sample Executable Program

A sample program, `Assignment3.exe`, is available for download at the course Web site. To save and run this program on your computer, you will need to grant permission when requested (as was done with the sample program for Assignment 2).

Programming Guidelines

(1) Use an array of structs for **each** of the following:

- To store the words of the dictionary selected in Option 1.
- To store the information on the unique words found in the passage.
- To store the words with the spelling errors.

(2) Only C-strings are allowed in this assignment. The *string* type should not be used.

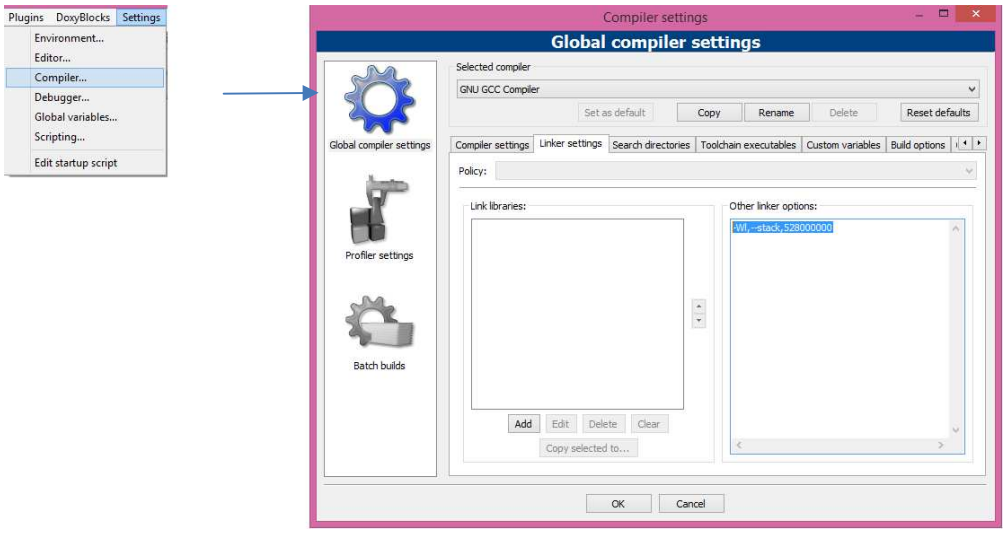
(3) You could write a function as follows to read the words from any one of the dictionaries:

```
int readWordsDictionary(Word words[], char language);
```

The *words[]* array of *Word* structs is passed to the function as a parameter (similar to the load functions in Assignment #2). The *language* parameter is a character corresponding to the language selected by the user ('E', 'F', 'S').

This function opens the appropriate dictionary file and stores the word in the dictionary array. It returns the amount of words in the dictionary or -1 if the dictionary was not created.

(4) Due to the size of the dictionaries to be loaded, a change has to be made to the CodeBlocks programming environment:



Click on Settings

→ Compiler…

→ Linker settings

Enter **-Wl,--stack,528000000**

in the box labelled "Other linker options"

(5) There are two functions which need to be use in the program: `strcpy` and `strcmp`.

(a) Both functions require you to include cstring that in at the top of the program the following line of code must be inserted `#include <cstring>`

(b) `strcpy` copies the C string pointed by *source* into the array pointed by *destination*, including the terminating null character (and stopping at that point). For example

```
char source[ ] ="Hello";

char destination [10];

strcpy(destination, source);
```

This copies Hello (including the termination null character into destination)

The size of the array pointed by destination shall be long enough to contain the same C string as source (including the terminating null character)

(c) `strcmp` compares the C string *str1* to the C string *str2*. This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.

`strcmp` returns an integer value indicating the relationship between the strings:

return value	indicates
<0	the first character that does not match has a lower value in *ptr1* than in *ptr2*
0	the contents of both strings are equal
>0	the first character that does not match has a greater value in *ptr1* than in *ptr2*

For example

```
char str1[ ] = "apple;

char str2[ ] = "orange";

strcmp(str1, str2) would return a value <0;

strcmp(str2, str1) would return a value >0
```