Programming Assignment #6

This assignment will give you more practice with file I/O as well as more practice with strings, arrays, and pointers. The main purpose of the program is to implement a rudimentary spell-checker. Besides checking words for correct spelling, you will also implement a few other functions. There are five functions that you need to implement to complete the assignment.

Function	Description
`char *` `mystrupr(char *string);`	Given a string, convert all lowercase letters to uppercase. Returns a pointer to the first character of the string that was passed in.

`int`
`words_starting_with(const char *dictionary,`
`char letter);`
	Given the filename of a dictionary, count the number of words that start with `letter`. If the file can't be opened, return `FILE_ERR_OPEN`, otherwise return the number of words that start with `letter`.

`int`
`word_lengths(const char *dictionary,`
`int lengths[], int count);`
	Given the filename of a dictionary, count the number of words that have length 1 to `count` and store them in the array `lengths` at the appropriate index. If the file can't be opened, return `FILE_ERR_OPEN`, otherwise return `FILE_OK`.

`int`
`info(const char *dictionary,`
`struct DICTIONARY_INFO *dinfo);`
	Given the filename of a dictionary, return some information about it (using the `DICTIONARY_INFO` structure). If the file can't be opened, return `FILE_ERR_OPEN`, otherwise return `FILE_OK`.

`int`
`spell_check(const char *dictionary,`
`const char *word);`
	Given the filename of a dictionary and a word, lookup the word in the dictionary. If the word was found, return `WORD_OK`. If the word was not found, return `WORD_BAD`. If the dictionary file can't be opened, return `FILE_ERR_OPEN`.

Notes:

- All of the functions (except `mystrupr`) need to open a dictionary. If it fails when attempting to open the file, you must simply return `FILE_ERR_OPEN`.
- There is exactly one word per line in the dictionaries. All words end with a newline character which is not to be included in the word (i.e. after reading the word from the file, you need to remove the newline from the word).
- You must include `stdio.h` in **`spellcheck.c`**, since you are using files. You should also include `string.h`. Both of these are included in the file for you already. You must not include any other files. If you do, you will receive a 0 on the assignment.
- The function, `mystrupr`, modifies the string in-place. In other words, do not make a copy of the string; you are modifying the string that was passed. This should be obvious to you by now by looking at the type (non-const) of the parameter. The return is the same value that was passed into the function. (like `strcpy` and `strcat`)
- The functions `words_starting_with` and `spellcheck` are case-insensitive when looking for characters or words. You need to account for this. One way to do this is to make everything uppercase or lowercase before comparing. The test driver shows examples of how your code should function.
- You should work on the functions in the order listed, since they are listed roughly by the order of difficulty.
- Remember to close your files when you are done with them. You will lose points if you don't. Not closing files is a serious resource leak in your programs. Please pay attention to your conditions which may cause you to not close a file. The compiler will not warn you if you fail to close any files.
- There are additional tips listed on the web page for this assignment. Before you start coding, **PLEASE READ THEM!**

Details
You are given a file called **`driver.c`**, which includes the main function with several test cases for you to use. You are also given a header file called **`spellcheck.h`** and a .c file called **`spellcheck.c`**. You are not to change the header file in anyway. There are five functions prototyped in the header file, one for each of the functions you are to write. As always, you must implement these functions *exactly as prototyped.* You will implement these functions in a file named **`spellcheck.c`**. A partial file is provided for you to start with. A sample command line to compile this assignment looks like this:

```
gcc -O -Wall -Wextra -Werror -ansi -pedantic driver.c spellcheck.c -o spell
```

This will compile and link both files and produce an executable file on Windows named **`spell.exe`**.

This assignment will only require you to include the two system header files mentioned. No other system header files may be included. Don't forget to put a comment next to each included file that lists the functions you are using from it. If you are not using any functions from `string.h`, then you should remove the include, otherwise, you'll need to add comments for it. All of the code that you write must be in **`spellcheck.c`**, since you will not be turning in any other files.

What to submit
You must submit **`spellcheck.c`** to the appropriate submission page.

Refer to the web page for this assignment for any additional details on how to submit this assignment. **Do not submit any other files than the ones listed.**

Files	Description
spellcheck.c	The C file. This file contains all of the source code for the program, properly formatted and commented as discussed in class.

If you've forgotten how to submit files, the details about how to submit are posted on the course website and in the syllabus. Failure to follow the instructions will result in a poor score on the assignment.

Make sure your name and other info is on all documents.