Assignment 2 - Spelling Correction

Due Nov 7 by 11:59pm **Points** 100 **Submitting** a file upload

Objective

Gain some practice in the use of strings, lists, files and functions in the C programming language.

Submitting Your Work

Submit your C program file and ***report*** (using the standard format) via the Assignment 2 **Submit Assignment** link on the course web site.

Introduction

In this assignment, you will write a C program to suggest spelling corrections to English words entered from the keyboard. Words entered by the user of the program will be compared to a list of over 105,000 words supplied in a data file. If an entered word is not found in the word list, your program is to suggest, as alternative spellings, those words from the word list that have:

1. The same length as the entered word, and
2. Minimum Hamming distance from the entered word.

Hamming Distance

The Hamming distance between two strings **of equal length** is the number of positions in the strings at which the corresponding characters are different. For example, the strings "ample" and "apple" have a Hamming distance of 1, since they only differ in the 2^{nd} character. The words "apfel" and "apple" have a Hamming distance of 3, since they differ in the 3^{rd}, $4^{th}$and 5^{th} characters.

Assumptions

You may assume the following:

1. Your program will not need to handle any words with length greater than 40 characters (including allowance for the terminating NULL character).
2. Where there are multiple words with the same minimum Hamming distance from the entered word, your program needs to report only the first 5 (in alphabetic order).
3. The word list file contains less than 110,000 words. Your program should read all the words into an array of 110,000 strings, each 40 characters long. You can declare this array as:

```
#define MAX_WORD_LEN 40
#define MAX_WORDS 110000

char wordlist[MAX_WORDS][MAX_WORD_LEN];
```

Then you can access the i^{th} word in the list as wordlist[i].

What your program must do

Your program must perform the following tasks:

1. Check whether the name of the file containing the word list is provided on the command line. If it is not on the command line, your program must terminate with an error message. The word list file txt is available on the course web site.
2. Open the file to read the words and add them to a list. There is one word per line. All words are in lower-case. Each line has the newline character '\n' at the end – you will need to remove this.

To open the file for input, use the fopen() function (declared in stdio.h):

```
int main(int argc, char *argv[]) {
   FILE *input = fopen(argv[1], “r”);
   . . .
}
```

You should use the fgets() function to read each line in the file:

```
fgets(wordlist[i], MAX_WORD_LEN, input);
```

3. Prompt for a word to check, using fgets() to read each word entered from the keyboard (stdin) by the user of the program:

```
char word[MAX_WORD_LEN];
fgets(word, MAX_WORD_LEN, stdin);
```

4. Convert the entered word to lower-case and either:
   - Confirm that is correctly spelled (since it was found in the word list), or
   - Suggest one or more words from the word list as alternatives.
5. Keep prompting for a word until the user inputs a word of length zero.
6. Use functions for (at least) the following tasks:
   - Load the word-list from the supplied data file.
   - Determine the Hamming distance between two words. Pass the two words as parameters to the function and have the function return their Hamming distance as an integer.

For full points on this lab exercise, your program must perform as specified above and conform to the coding standards.

Coding Standards

1. Use meaningful names that give the reader a clue as to the purpose of the thing being named.
2. Avoid the repeated use of numeric constants. For any numeric constants used in your program, assign their value to a variable and then use that variable wherever the value is needed.
3. Use comments at the start of the program to identify the purpose of the program, the author and the date written.
4. Use comments at the start of each function to describe the purpose of the procedure and the purpose of each parameter to the procedure and a description of its return value (if any).
5. Use comments at the start of each section of the program to explain what that part of the program does.
6. Use consistent indentation.

Hints

The word lists are in the files section. There are two - wordsEn and wordlist. One is smaller and easier to use for initial testing.

Bonus Section

There are extra credit extensions to the exercise, for up to an additional 20 points.

1) Extend the process of finding alternative words to include those whose length differ from the length of the entered word by 1 character. This must include words that are one character longer or shorter than the entered word.

In comparing the entered word to word-list candidates that are one character longer, you must try all alternatives achieved by deleting one character from the candidate word. When comparing the entered word to word-list candidates that are one character shorter, you must try all alternatives achieved by deleting one character from the entered word. In all such cases you need to add 1 to the Hamming distance calculated from the comparison of the resulting equal-length words. (5 bonus points)

2) Create alternative methods for measuring the distance between words, for example weight the Hamming distance by different criteria, such as: 1 for vowel to vowel or consonant to consonant differences and 2 for vowel/consonant differences; weight by the alphabetical distance (possibly as a fraction); weight by the distance on a qwerty keyboard; or specify your own reasonable criteria. Compare the quality of any added weighting to the standard Hamming distance. (5 bonus points per enabled criteria)

3) Create an alternative language dictionary and test (5 bonus points).

4) Create, test, and explain your own extension (5 bonus points).

5) As with any dictionary, there will be words that are inappropriate for use in many places, but which remain in the dictionary. If a student doesn't wish for any particular word, or set of words, to appear as an alternative for a misspelled word, then they are free to create a function that prevents their selection (5 bonus points).