CSC112 Lab 4 – Counting Words Spring 2016

In this lab, we will scan an input text and count the number of times each word appears.

Input: A text file containing English sentences.

Output: A list of words and how often each word occurs For example:

```
apple     3
carrot    1
grape     7
pear      2
```

In C and C++ a character string can be represented as an array of characters in which the end of the string is marked by the special ASCII zero character. In C and C++, the ASCII zero character (sometimes called the "null character") is written `'\0'`. When passing a string to a function, it is usually passed as a pointer to a character.

The stream operator `>>` can be used with a fixed-length array of characters to read each word. For example:

```
const int MAXLEN = 64 ;
char aword[ MAXLEN ] ;

cin >> aword ;
```

The stream operator will ignore spaces until it finds a non-blank character. It will then copy all non-blank characters into the array `aword` until it finds the next blank. This is an easy way to break the input stream into words, but it has a drawback: some words have punctuation at the end, and that punctuation will be included in the "word". When counting words we don't want to treat "book" and "book." as two different words. To resolve this problem, loop through each position in the character array, and cut the string short by replacing any punctuation character with the null character. For our purposes here, check for: comma, semi-colon, colon, period, exclamation mark, hyphen, and question mark. You will notice there are places in the text with two hyphens "`--`". When you replace the first hyphen with a null character, you will be left with a word of length zero. Do not add words of length zero to your list of words.

Also, we do not want differences in capitalization to count as distinct words; i.e., "Fun" and "fun" should be counted as the same word. To accomplish this, loop through every character in the string and convert every upper case letter to lower case. If a letter is between 'A' and 'Z' then add the quantity ('a' - 'A') to that letter. Notice that characters are compatible with integers, and you can do arithmetic with the ASCII codes.

To process the incoming stream of words, you will need two arrays and one counter:

- one array of character strings to hold your list of words
- one array of integers to hold a counter for each word
- one counter to keep track of how many words are added to the list.

Suggested global constants:

```
const int MAXLEN=64     ;  // Maximum length of one word.
const int MAXWORDS=4096 ;  // Maximum number of words
```

Suggested declarations in the main program:

```
char aword[MAXLEN] ;       // Array to hold one word.
char * wlist[ MAXWORDS ] ; // Array of strings to hold all words
int num_words  ;           // Number of words currently in wlist.
int wcounts[ MAXWORDS ]  ; // Number of times each word occurs
```

Suggestions for implementation:

- Write a function called `find`. Input parameters include `aword`, `num_words`, and `wlist`. The function should return the position in which `aword` is found in `wlist`. Return -1 if the word is not found.
  - Writing `find` requires you to compare two words. Use the library function `strcmp()` to compare two strings. Remember `#include <cstring>`
- As you input each word, use your function `find` to see if it is already in your word list. If so, increment the corresponding counter in the appropriate position of your `wcounts` array. If the word is not in the list, put it at the end of the word list, and set the corresponding counter to 1.
- As you put each word at the end of the `wlist` array, be sure to check that there still is room. I.e., if `num_words` reaches `MAXWORDS`, there is no space left. If that is the case, give an error message and exit.
- When you want to insert a new word at the end of the array `wlist`, you can not make a simple assignment. Even if that worked, the entry on word list would be pointing to a block of memory which would be over-written on the next word. Use the library function `strdup()` to make a copy of the input word. E.g.,

```
wlist[num_words] = strdup( aword ) ;
num_words++ ;
```

Running Your Program

Download the file `sample_text` from menehune.opt.wfu.edu. Run your program with:

```
gottlieb% a.out < sample_text > word_counts
```

Turn In;

Save all your work in a directory named "Lab4". Change to your home directory (the parent directory of "Lab4"), and create a file named "lab4.tar" using the command:

```
tar cf lab4.tar Lab4
```

Use `sftp` to upload the file "lab4.tar" to your account on telesto.