

Due date: Friday, November 19, 2010

1 Purpose

In this project you will write a program that will perform a "word count" of the text file directed in. It will create a list of words that are in the file and their corresponding counts. A word is defined as a space delimited sequence of characters. In the words stored, all punctuation will be removed and all uppercase characters will have been replaced by their corresponding lowercase characters.

The project will maintain two arrays one which contains all of the words found in the file being ready (words as defined above) and the other will maintain the corresponding counts of how many times that word appeared in the input. For example, if the file contains only "this is my this" the word array would contain {"this", "is", "my"} and the count array would contain {2,1,1} after all of the input is read.

2 Project description

As you read this section, look at the input and output files for the proj5aMain.c and the public tests all included.

The input to your program is to consist of a sequence of lines. No line will be longer than 80 characters and there will be no spaces at the beginning and end of the lines. Each word in the file will be separated from the next word by a single space character. There will be no blank lines in the file, and there will be no more than MAX distinct words.

You must implement each of the functions as described in the `proj5a.h` file. You may also add additional functions as helpers for these, but these helpers do not get prototypes in the `proj5a.h` file so they can not be called from the files that contain the main functions.

The main (provided) creates two empty arrays, reads lines of input adding any new words to the array and updating any of the element of the count array to make the arrays accurately reflect what has been read. As you fill these arrays, there is a `size` integer that is always passed so that you know how many Strings and corresponding integers are currently stored in the array.

When the array contents are printed. They are always printed with a count at the top to indicate how many values are currently stored in the array. This is followed by each of the elements on its own line.

3 Provided Files

In your `tgz` file, you have:

1. a `proj5a.h` file that includes all of the header files you will need. It also defines constants and types that make your implementation easier and prototypes that make your implementation possible.
2. a `proj5a.c` file which is a skeleton for you to fill in the code in the required functions. Do not change any of the function headers provided there since they match the corresponding prototypes.
3. several `public0?.c` files which each contain a main method designed to test one (or more) of the individual functions you are writing. Each of these has a corresponding `public0?.output` file which shows the output that you should get if your functions are implemented correctly. These files also have their corresponding `public.in` file if there is input required for that test. You should implement one function in your `proj5a.c` file and then compile that `proj5a.c` file with the appropriate `public0?.c` file to test that the function you just finished works correctly before going on to the next function. You can also write additional test files using these as a model in order to test other functions you write.
4. a `proj5aMain.c` file which should not be modified. When you have finished testing all of the functions individually, you can compile your `proj5a.c` file with the `proj5aMain.c` and run it with the files provided that are named `input?`. You should get the output shown in the corresponding `output?` file.

4 Project Requirements

All your C programs in this course should be written in ANSI C, which means they must compile and run correctly with `gcc -Wall -Werror -ansi -pedantic-errors` on the grace system as was setup at the beginning of the course. You will lose credit if your program generates any warning messages when it is compiled. Even if you already know what they are, you may not use any C language features other than those introduced so far in lecture or lab. Note specifically that structs may not be used. In addition, neither the `goto` nor the `continue` statement may be used, and the `break` statement may not be used in any loop. Using C features not presented in the lectures before the date of the assignment, or using the `goto` statement, or `break` or `continue` in loops **will result** in losing credit. You may use any of the `string.h` functions and any of the `ctype.h` functions already defined in those libraries. Many of these have been discussed in class, and you do not need to use any beyond what has been discussed in class.

You may not modify the `proj5a.h` file at all. You may not modify the `proj5aMain.c` file at all but you are encouraged to create others based on it as a model. All of your code must appear in the `proj5.c` file. You may add other methods or constants to the `proj5a.c` if you would like, but they will not have a prototype in the `proj5a.h`.

All work for this project must be your own individual work. You may not work with any other student or compare code in any way. If you need help see either the instructor or TA. Obtaining help from others will be considered cheating. Getting code from a web page or other location will be considered plagiarism.

Your program must have a comment near the top which contains your name, login ID, student ID, your section number, and an original description of the action and operation of the program. At the end of this comment block, you must also include the honesty statement: “I pledge on my honor that I have not given or received any unauthorized assistance on this assignment.”

Your program should be written using good programming style and formatting, as discussed in class and throughout your textbook. For this project, style is considered to consist of:

- You must implement each of the functions defined in the `sudoku.c` file without changing the name, return type, parameter(s) or what it does.
- adequate descriptive comments throughout (in addition to the comment mentioned above), to describe what your program is doing and how
- neat and proper indentation and formatting, including consistently and correctly aligning braces using any of the bracing styles illustrated in your textbook
- writing clear and readable code
- use of meaningful descriptive variable names
- use of symbolic constants for all “special” values which don’t change and which are used in more than one place in your code
- avoiding disallowed C language features as discussed above
- make sure it runs with the sample input/output files given before submitting it, and make sure you test it on others that are different from those given.