CPSC 327
Project 2

Assignment

Write a basic word reference which can do the following;

1. Open a file for reading.
2. Read all the words from the file, count the number of times each word occurs and records this data using an array of structs.
3. Alphabetize the array of structs (A-Z)
4. Write the alphabetized array of structs to an output file.

You must use an array to hold your list (NOT A VECTOR). You may use std::string.

Where to start:
Define all the functions in array_functions.cpp (no implementation yet) just enough to get it to compile, then run it. You will then see the output of running all tests plus your grade.
Then start filling in the implementation details.

Helpful Bits
ONLY MAKE CHANGES TO array_functions.cpp. THAT IS THE ONLY FILE OF YOURS THAT I WILL TEST

- See the starter project Project2_Starter. I have given you quite a bit of code, you have to fill in datastructures and function definitions in 1 file, array_functions.cpp. Use the declarations in ,array_functions.h as a guide to what functions. You are on your own in terms of the datastructures with the provision that you must use an array.
- to turn a std::string into a const std::string use c_str(), you probably need this for opening files. Example stringname.c_str()
- make sure you call strip_unwanted_chars to get rid of the rubbish in a token
- Use stringstream to parse each line. Here is a bit of code that may help:

```
:
#include <sstream>

void extractTokensFromLine(std::string &myString) {
```

```
        stringstream ss(myString);
        string tempToken;

        while (getline(ss, tempToken, CHAR_TO_SEARCH_FOR)) {
                processToken(tempToken);
        }

}
```

This function takes myString and searches for tokens separated by constants::CHAR_TO_SEARCH_FOR (a space). This constant is defined in file constants.h. For each token it finds it calls ProcessToken(tempToken).

- Please use a struct like this to hold each index entry. Useful for tracking a token and the number of times the token occurs.

```
struct entry
{
  string word;
  int number_occurences;
}; // REMEMBER the last semi-colon
```

- For sorting you are mostly on your own. Consider this however. Unlike Java, strings can be compared with >, <, and ==. Use this fact to alphabetize your list.

Testing and Verification

Project2.cpp is a tester, it runs various tests on your code (defined in array_functions.cpp) and will use the test results to calculate your grade.

Sample Run

See ./data folder in the sample project. testdata_full contains some words, testdata_full_processed is correct output after running project2.cpp's test_file function with no sorting. Similarly testdata_full_processed_sorted is with sorting.

To Turn In

I just want your array_functions.cpp file, nothing else. Please do not zip this file. Please do not change any of the .cpp or .h files I give you, please do not add any files that your array_functions.cpp depends on since I will not have access to them.

Scary parts

I'm using a templates in project2.cpp. Its sorta like a Java generic, Don't worry about it. Its there to condense code.

I'm also using popen in project2.cpp as a way to run another process, in this case some linux cleanup code and a diff tool. The diff tool that I am using is the standard one on a linux install. I've noted its details in the code.

Grading
Points awarded as per Project2.cpp output (as long as functions attempted).

Special cases:
-5 turn in more than array_functions.cpp
-100 does not compile