CS 332/532 – 1G- Systems Programming

HW 1

Deadline: 01.30.2022 Sunday 11:59pm

Objectives

To implement a C program that uses structures, dynamic memory allocation, and string functions provided by the C library.

Description

The goal of this project is to implement a C program that a variable number of keywords as command line argument, reads text from the standard input stream, searches the keywords in the input stream, and when the end of input stream is reached prints the number of times each keyword appears in the input text.

Here is a sample input and output (input is in *italics*):

```
$ ./a.out an and or to

This is some sample text. You have to find out how many times certain keyword
s appear.
It is possible that some words may not appear at all while some words might a
ppear multiple times.

There could be newlines and white spaces and long lines. You should keep acce
pting the text until the end of file character is found. You can use the exam
ple program getline.c to figure out how to read line by line until there are
no more characters in the input stream.
^D
Here is the number of times each keyword appears:
an: 0
and: 2
or: 0
to: 3
```

Note that the keywords should match exactly and are case sensitive. For example: “an” is part of “many” and “can” but there is no explicit “an” anywhere in the text.

Guidelines and Hints

1. Design the program using separate functions to perform each of the key tasks. For example:
    1. The first task is to read the keywords provided as command-line arguments and create a table with the keywords and counts. Use a structure with two entries: one for the keyword and one for the count and then create an array of structures based on the number of keywords. Note that you should not use static declaration and hardcode the number of keywords, the program should take variable number of keywords. Initialize the count field to zero to start. You can create a function to perform this task, say, init_table with appropriate arguments.
    2. Next read the text from standard input using the getline() method provided by the C library as shown in the example getline.c (you are free to use any other C library function that you prefer).
    3. Since we have to compare the keywords with complete words (not partial matches) we have to break down the line we read into keywords. You can do this using a separate function or use one of the C library functions. Look at [https://en.cppreference.com/w/c/string/byte (Links to an external site.)](https://en.cppreference.com/w/c/string/byte) for a complete list of string functions provided by the C library.
    4. After you read a line, search for the keywords in the input and update the table if a match is found. You can use one of the C string library functions to perform the search. You can create a function to perform the search and update the table, say, update_table with appropriate arguments.
    5. When you reach the end of the input stream, print the table. You can use a separate function to print the table, say, display_table with appropriate arguments.
2. Test the program with simple input as shown in the example above. Instead of typing the input text every time, you can save the sample text in a file (say, input.txt) and use I/O redirection as shown below:

```
$ ./a.out an and or to < input.txt
```

3. Once you have tested your program with simple input files, you can test the program with larger text files (tintTale.txt, tale.txt) available in Canvas in the Project directory using the I/O redirection (as shown in step 2 above).
4. Comment you code and include a comment header section that provides instructions for compiling and running the program.
5. You have to test this program on the CS Linux Systems (the Vulcan machines). If the TA is not able to compile and run your program on the CS Linux Systems you will not get credit for this project.
6. Upload only the source file(s) to Canvas.

Submission

Upload the C source code, PDF version of C source code, and README.txt (or readme.md) file to Canvas. Do not include any executable or object files.

Use the following command to create PDFs of your source code. (replace the <Source_code_File_name> and <Output_Source_code_File_name> with your C source code file name):

```
$enscript <Source_code_File_Name>.c -o - | ps2pdf - <Output_Source_code_File_Name>.pdf
```

Grading Rubrics

Description	**Points**
Creating a dynamic table (array of structs) based on the command-line arguments	20
Reading text from standard input	10
Tokenizing the text and creating a dynamic array	20
Comparing keywords and updating keyword count	15
Display the keyword/count table	10
Use of suitable functions to perform the different tasks above	15
Documentation including README file	10