CSE 5462: Lab1 (100 points)

Demo in Class or by appointment: January 21
Electronic Code Submit Deadline: 9pm, January 21

Important: We will demo in class or in lab. For in class demos, we will go around the room. Be prepared to demo at the start of class. The submitted code will be used only to verify that you did not copy from others, to compile and re-run your program, to make sure you were indeed demonstrating your own code, and to grade for documentation of your code.

Write a program in C called "count" to read a binary file and print the following statistics on the screen as well to an output file:

- the size of the file in bytes
- number of times the search-string specified in the second argument appeared in the file

You will run the program using the following command:

count <input-filename> <search-string> <output-filename>

The string matching algorithm does not have to be the most efficient in time.
If the input-filename is incorrect, or the number of arguments is incorrect, or the output-file cannot be created, the program must print appropriate messages and show how to correctly invoke it.
The search string will be 20 bytes or less. The search string and the file both may contain binary characters.

The input file can be of any arbitrary length. The implication is that you cannot have the entire file in the memory at any point of time. You will read the file in chunks of 100 bytes or smaller. Note that you also have to handle the case when your search string starts in one chunk and ends in the next chunk.
If the output file already exists, it will overwrite that file.
During execution, any array in the program or malloc-ed space must not exceed 100 bytes.

Submit well-documented and well indented code along with a README file explaining how to run the program, and a makefile. Submit it using GitHub, in a subdirectory called Lab1

The grading rubric is as follows:

- Program correctness and robustness: 80%
- Coding style (comments, indentations, README, Makefile): 20%

What you should get from this lab?
Tools:

- basic file IO (fopen(), fread(), fwrite(), fclose(), ftell(), fseek(),fflush() etc);
- basic string manipulation (memcpy(), memset(), etc);
- parameter passing (by value and by reference)

Concepts:

- how to handle files (input and output) – this is useful and analogous for socket programming;
- how to properly handle return codes from function/system calls;
- how to write logic to do multi-pass scans on files/data structures;
- tradeoffs between memory usage and processing time;
- when is data actually written to disk and why do you have to worry about that:
- how to process subsets of a file/structure when it can't all be read into memory;
- how does the computer treat bytes (binary versus text), when and why that matters;
- how to write a solid C program, including when/where to use subroutines, when to use pass by reference and when to use pass by value, how to compile a program, how to debug it, etc.
- Start thinking about how you would make the server 'loop' and process multiple files….from multiple clients (serially)