SCC110: Software Development Term 1. C Programming.

Problem Set 8: Strings and File I/O

The Goal
The processing of files is critical to most applications computers can be put to. Whether searching for patterns while sequencing the genome, analysing images from the Hubble space telescope, screening mammograms, or more prosaically, storing our emails, Word documents, family photos or the databases behind Facebook. It's all just files and file Input/Output.

Your task today is to take on the role of a computer security and forensics expert again (more CSI 110 ;-)) and to write a program that can analyse a log of people accessing a web site. We have provided you with an 'access.log' file from a web server. Your task is to read in each line of the file (corresponding to a 'hit' on a page of the web site) and to gather some statistics by looking for patterns in this data.

1.1 Log Analysis

Incrementally develop a solution that:

1. How many 'hits' are there in total in the log? (*Hint: there is one 'hit' per line of the file*)
2. We think our attacker may be using the 'Mozilla browser', how many lines contain the identifying fingerprint 'Mozilla'?
3. We're seeing a lot of 'possible attacks' coming from addresses starting with (lines starting with) 66.249.66—can you print out every page they access? (*you only need print the line unless you want to go further to test yourself*) How many accesses do they make in total?

You should think carefully about your algorithm and approach on paper first. Don't forget good coding practices, including choice of meaningful variable names, comments and indentation.

Hacker edition
1. How many unique addresses do we see? (hint: look at the IP address at the start of each line-these look like '148.88.61.11')
2. Can you tabulate how many hits from each address? (hint: you'll need an array for this). *Optional: Use a dynamic/resizing array or linked list for this.*
3. Based on this, should we fear hits from 66.249.66.*?

As before, you should be looking to make your code a) great to read, and b) solve the problem elegantly. Be sure to ask our opinion on your code in the labs.

Resources
As part of today's problem set you will get more familiar with how to do file processing in C. You will need to use the following functions from **#include** <stdio.h>:

- `FILE *fopen(char *filename, char *mode)` —Open a file with mode 'mode'. See man `fopen`, the example below or use a web search for more details and examples!
- `char *fgets(char *str, int size, FILE *stream)` —Get a line from the file (stream) and copy it into character array '`str`'. Take care that the value for 'size' is large enough to *read the whole line* of a 'hit' in the access log. *A very common error is not to declare an array big enough (and thus allocate enough space) for the resulting data!*
- `int fclose(FILE*stream)` —Close file'stream'. This is a variable containing the *file handle*, not the filename!

You may also find it useful to explore **#include** <string.h> in more detail:

- `int strncmp(char *s1, char *s2, int n)` —Compare the first ‘n’ characters of strings ‘s1’ and ‘s2’.
- `char *strchr(char *s, int c)` —Find character ‘c’ in string ‘s’ and return a pointer to it (or NULL if it’s not found).
- `char *strstr(char *s1, char *s2)` —Find string ‘s2’ in string ‘s1’ and return a pointer to it (or the special value NULL if it’s not found).

Please also *revisit last week’s lecture* covering arrays and strings.