Assignment 5

Your goal in this assignment is once again not to implement any major new code, but to build upon last week's in-lab exercise to use hashes in order to solve an I/O or file systems' related problem. In addition to the three programs and README file mentioned below, you are to submit the key-value code you implemented in last week's pre-lab (**50 points**).

You are writing code for a new kind of computer peripheral intended for a special kind of I/O. Specifically, this device behaves like an append-only never-to-be-modified file. Your software for this device needs to accept arbitrary sequences of bytes (and so you should be able to accept input strings), and then stores these strings for future evaluation. The problem is that we cannot be sure if someone has tampered with the device/file in which we are placing these strings, and you are unable to remember the full sequence of strings that you sent to it. And so, you need to use a hash function to include with every string you write a code at the end (the result of the hash) that allows you to check that the sequence you saved is the same one that you wrote originally.

To this end, you are to write and submit **three** programs for this lab (**100 points**). The **first** should accept sequences of strings until an end of file is detected, and output the following (do NOT start a new line between each item):

- the length of the string (hint: please take care to print this number out carefully)
- the string itself
- the result of applying the hash function on the string.
- the result of applying the hash function on the string and the preceding hash value (i.e., concatenating the string with the hash value, where the very first string assumes the preceding hash was simply the null string).

You should run this program and save the output to a file (you should use only Unix command line options to achieve this result, in other words, the output of your program should be sent to stdout, but a redirect ">" should be used to pipe the output into a file).

The **second** program should take any such file as its input, and should only print those strings in the file that do not have a valid hash code.

The **third** program should take any such file as its input, and should print out a message indicating if it has detected an omission from the file (i.e., are there any missing strings that it could detect?).

In addition to these three programs, you should also provide in your README an explanation of your best answer to the following **two questions** (**50 points**):

1. How would you use such a program to detect duplicate strings (without modifying the programs themselves, but only using operations conducted on their output)?
2. How would you save space, using the key-lookup that you have previously implemented, if you were told that the strings were very long, and that many of them were identical to each other?

Good Luck!