Project 3: Password Blacklist

Submission deadline: 9pm on 11/23, 2023
(9% of the total grade + 1% extra credit)

Imagine you work for an online service provider. Due to a recent password leak, you have been asked to create a blacklist (of size n) of commonly selected passwords so that users can be restricted from using such insecure passwords. Write a program that reads a stream of leaked passwords (one per line) from stdin and prints the frequency of each word in the sorted order (from the largest to the *i*-th largest). The program takes '*i*', the size of the blacklist, as an optional command line argument. So, if *i* is 5 (as below), your program must print the most frequently selected 5 passwords by their frequency in the sorted order. If *i* is missing, your program should print all words from the largest frequency to the smallest frequency.

```
./blacklist 5 < ./leakedPasswordList.txt
The 5 most commonly occurring passwords are:
2650    123456
1244    password
708     phpbb
562     qwerty
418     12345
```

Requirements & Information:

- Write the code in C – your code should compile with gcc installed on eelab5 or eelab6
- The input text consists of an arbitrary number of passwords (containing various characters, numbers, and symbols). You can assume all passwords are valid – you should be sure to read the full string on each line and cull the trailing eol.
- Check the correctness of your program by comparing the result of:
  `sort ./ leakedPasswordList.txt | uniq –c | sort –k 1 –r –n | more`
- We provide a text file for testing. Please see the attached file.
- You can assume that the same password does not show up more than 2 billion times.
- You can use standard C runtime libraries (including qsort()) but you cannot use any library that implements basic/advanced data structures (like a hash table, a priority queue/heap, etc.)
- Name your source code file as blacklist.c.
- You will get **ZERO** point if the source code doesn't compile (due to compiling errors). Make sure that your code compiles well before submission.
- Suppress all warnings at compiling by turning on the "-W -Wall" options in the CFLAGS in Makefile. You will get **some penalty** if gcc produces *any* warnings.

Collaboration policy:

- You can discuss algorithms and implementation strategies with other students, but you should write the code on your own. That is, you should NOT look at anyone else' code, including viewing online source code to solve the problem or asking LLMs to generate source code.

Submission:

- You need to submit three files – "blacklist.c", "Makefile", and "readme" in a tarred gzipped file with the name as YourID.tar.gz. If your student ID is 20231234, then 20231234.tar.gz should be the file name.
- 'make' should build the binary, "blacklist".
- "readme" (a text file with "readme" as the file name) should describe how you implement the code. (1) Explain the data structure you chose to implement and the overall algorithm. What is the worst-case running time of your algorithm in terms of Big-O notation? (2) Explain how you checked the correctness of your program (3) Add some timing results for the password file we provide.

- **Extra credit up to 1%.** Extended your program to count the occurrence of *n-grams*, or substrings of particular lengths (n), in addition to whole passwords. For example, with n=4, all n-grams for the password "qwerty" are "qwer", "wert", "erty", while with n=3, they are "qwe", "wer", "ert" and "rty". Such frequency counts of substrings are useful for generating probabilistic models of passwords (very much optional: see https://www.ndss-symposium.org/wp-content/uploads/2017/09/06_3.pdf if you want an example use case). Your code should handle this task by accepting a second optional parameter, with the length of the substrings to calculate. For example, the following would produce a list of the top 4 most frequently selected four-character substrings. If no second parameter is provided, the code should run as normal for the main assignment.

```
./blacklist 5 5 < ./leakedPasswordList.txt
The 5 most commonly occurring 5 character substrings are:
4430    12345
3788    23456
2582    phpbb
1708    sword
1667    passw
```

Grading:

- We will evaluate the correctness of your program.
- We will give a penalty to a program that is excessively slow – if you chose the right data structure/implementation/algorithm, it should work fine.
- We will evaluate your readme file.
- We will evaluate the clarity of your code – coding style, comments, etc.