**User’s Manual**

**Description:**

This program uses lastNames.txt to generate two files: raw.txt and encrypted.txt. The raw file hosts users last names and their unencrypted passwords, while the encrypted file hosts users last names, and their encrypted password. When run, the program will take the first five lines of each file and compare the passwords; first checking raw directly against encrypted, then the raw password will have one character altered and be checked against the encrypted password. At the end of the run, the user should see five “MATCH” entries output on the screen and five “XXXXX” or unmatched entries on the screen.

**Output Format:**

“MATCH” indicates a raw password matched the encrypted password when deciphered.

“XXXXX” indicates a raw password did NOT match the encrypted password when deciphered.

LEGAL PAIRS:

UserId Password Result

smith aaaaaaaaa MATCH

johnson bbbbbbbbb MATCH

williams ccccccccc MATCH

jones ddddddddd MATCH

brown eeeeeeeee MATCH

ILLEGAL PAIRS:

UserId Password Result

smith aaaaaaaab XXXXX

johnson bbbbbbbbc XXXXX

williams ccccccccd XXXXX

jones dddddddde XXXXX

brown eeeeeeeef XXXXX

**How to Run:**

* Load folder into PuTTY and access host directory (should be “digioacchinoe”)
* Use “make all” command to build the project.
* Run program using “./UserID”

If you wish to unit test the system before hand:

**Make testCypher** – outputs the encoded pair “jones” and “moge” and decoded pair “jones” and “data”– whose result should be “OUTPUT:\n moge\n data”

**Make testHash** – outputs the contents of the table specified in testHash.cpp. Contents should be organized into buckets by first character, and appends new elements at the end of the bucket. This method sorts the elements alphabetically and by priority.

**Make testFileRaw** – After this module is run, you should see two new files in the directory: raw.txt and encrypted.txt. The two files should share the same list of names but not the same passwords. On the screen the program will output that it is “DONE” and there are “26” buckets.

**Make testFoundation** – Output the contents of the generated hash table, from all data in encrypted.txt. The output will show that the buckets are organized by first letter, and elements by priority.

**Make all**

**testAll & UserID** – These options generate the same result as discussed in the description on page 1.

**Github Repo:** The github page for this program can be found at:

<https://github.com/EricDiGi/UserIdPassword.git>