CSC8501 Coursework – 2021
Collatz Password Manager
Due 22nd October 2021 at 16:30

Set by Graham.Morgan@ncl.ac.uk

$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \equiv 0 \pmod 2 \\ 3n+1 & \text{if } n \equiv 1 \pmod 2. \end{cases}$$

Specification (what you need to do): You will build a computer program in C++ to encrypt user passwords and authenticate user passwords. You will then extend your computer program to test the strength of user passwords. You will demonstrate your program (assuming you are able to attend campus) from 10am on 22nd in Game Lab (USB).

The rules governing password encryption and password storage

- We assume the use of the Collatz Conjecture in determining password encryption
  - Password characters are limited to the printable ASCII characters
  - Passwords can be any length (including 1)
  - A username can be any length
  - A password file stores usernames (in plain text) and encrypted passwords

Password file format (`password.txt`)

The password file stores usernames followed by a space and then the encrypted password. This is achieved each newline:

```
graham 12334532156745
gary 3422892347
rich 34790239487239
giacomo 344589009120
```

Encrypting and storing passwords

You allow users to create usernames and passwords and add them to the password file.

- Allow input of username and password by a user
- An offset value of 0 is initially set for the first character of the password
- Each character of the password is considered from left to right when encrypting
  - A character is turned into its ASCII decimal representation and offset added
  - This number is the seed for the algorithmic steps in the Collatz conjecture
  - Count the number of turns required for the algorithm to reach 1 for the first time
  - This number is the encrypted version of the character
  - This number becomes the new offset value for the next character
- Once all characters are encrypted you have a fully encrypted password

- This password is added to the password file together with the associated username

Authenticating passwords

You allow users to authenticate themselves via valid username and password combinations.

- Allow a user to enter their username and password
- Check that the username is present in the password file. If not then indicate “failure!”
- If username does exist
    - Allow the user three attempts to enter the password
    - Check if the password is correct
    - You are authenticating an unencrypted password against an encrypted password
    - If the correct password is entered then indicate to the user “success!”
    - If an incorrect password is entered then indicate to the user “failure!”

Password strength analysis

You must create 20000 random passwords and encrypt them for testing

- There is no need for a username now
- Save passwords generated randomly in the password test file (`passwordtest.txt`)
- Passwords are presented in the file one line at a time like so:

```
1283476239842
94568345235235634
982392356
```

- You are required to test combinatorial variance and length variance
    - For the first 10000 passwords repeated characters are allowed and ASCII values are restricted to 10 lowercase letters (randomly chosen by program)
    - For the second 10000 passwords repeated characters are not allowed and ASCII values may assume values from the whole extended ASCII set (256)
    - Both sets of 10000 passwords may be 1 to 100 characters in length:
        - 1 - 100 and 10001 - 10100 are 1 character in length
        - 101 - 200 and 10101 - 10200 are 2 characters in length
        - 201 - 300 and 10201 and 10300 are 3 characters in length
        - …
        - 9901 - 10000 and 19001 20000 are 100 characters in length
- You may choose any approach you wish to test the strength of the passwords
- Clearly describe how long it took to discover passwords (on average) for each category
- Clearly describe what percentage of passwords were not discovered for each category

Your program (hints and help)

Allow your program to present 4 options:

1. Create username/password
2. Check username and password
3. Generate password strength analysis file
4. Analys password strength analysis file

Use the `passwordtest.txt` file of your colleagues to see whose program can guess the most passwords the quickest.

Marks Available (100):

- **20 Marks for creating password entry**
    - **2** Marks for displaying the menu; **4** Marks for accepting username and password from user input; **2** Marks for opening a file and writing to it; **2** Marks for username and password creation in password.txt file; **10** Marks for successful creation of the encrypted password using the Collatz Conjecture;
- **20 Marks for password authentication**
    - **4** Marks for catching erroneous input; **6** Marks for catching file system exceptions (e.g., file missing); **2** Marks for allowing retries; **8** Marks for validating password correctly with appropriate retries
- **30 Marks for password strength analysis**
    - **8** Marks for generating 20000 passwords as described and placing in file; **10** Marks for successfully cracking any number of passwords; **6** Marks for describing the average time it took to crack the password for each category; **6** Marks for displaying percentage of passwords cracked per category
- **20 Marks for advanced features**
    - Student can pick up **2** Marks for each advanced feature from the following: *Classes, Inheritance, Polymorphism, Lambda Functions, Compile Time Optimizations; Pointers-to-Pointers; Templates; Assembler/Machine Code Augmentation; Pointers-to-Functions; Threaded Programming; Smart Pointers; Auto Keyword; Any Type*
- **10 Marks for decrypting this (a well known sentence in English):**
    - 27322810313331033910211452912207344136146925461033281533271031012815108114101

Submission

- You must submit your work in NESS (a complete version of your Visual Studio Project).