

**Name: Shawn Ronaldo Grant**

**Cand. #: 1000160537**

**Center #: 100016**

**Center: Campion College**

**ZENCYPHER**

**Text Encryption**

Computer Science Internal Assessment

**CAPE Computer Science Unit 1 Internal Assessment**

**TABLE of CONTENTS**

* Introduction
  + Background 2
  + Context Description 3
  + Problem Description 4
  + Objective Of Solution 5
* Requirements Specifications
  + Techniques of Analysis 6
  + System Models
    - Dataflow Diagrams 7
    - Entity Relation Diagram 9
  + Functional & Non-Functional Requirements 10
* Design Specification
  + System Architecture (HIPO chart) 11
  + Records and Data structure justification 12
  + Interface Specification 13
  + Algorithm/ pseudocode 14
  + Test Plans 22
* Application Development
  + C code 24
  + Test Plan Results 44
* Conclusion 48
* Appendix 49

**INTRODUCTION**

**BACKGROUND**

Privacy of information has been an important part of history, from messages been sent secretly by birds to now, where we use the internet to send most of our information. Information sent over the internet can include: usernames, passwords, addresses and telephone numbers**.** With the rise of the internet, new threats to data security have arrived. All computer systems are susceptible to intrusion by hackers. New technology has been developed to combat the threat of hackers, but no system is hacker-proof yet.

Everyday people send sensitive data over the internet via social media or email. Although these networks may be strong against computer based attacks, no thought is put into the fact that once your phone or laptop is left open anyone can simply open your messages and view the data you’ve sent to others in plain text.

This is where local encryption is useful. Text and files can be encoded, so that even with access to the device it is impossible for anyone to make sense of the information. There are many different encryption algorithms, most popular being: MD5, RSA, SHA-1 and 3DES.

**CONTEXT DESCRIPTION**

Encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot.

When people send information over the internet, messages are encrypted and sent over a network to the recipient’s router, where it is decrypted and returned to the recipient’s device and represented as a plain text or file.

This works well for ensuring that external hackers cannot get intelligible data from information sent over the network.

This encryption, however, does not protect against anyone with access to the device from viewing messages as the data is only encoded while it is in transit.

If, say, you text a friend or relative your email password (for whatever reason), and either of you gets robbed and your phone stolen. There’s no way of protecting that message that was sent. The thief now has your email. Even if there is a lock on the device, hackers have always figured out ways of cracking those, since they are normally not very complex.

If the message sent was in an encoded form it would add an extra layer of protection on the information. There would be no way of determining what the original message was.

**PROBLEM DESCRIPTION**

* We share sensitive information regularly over plain text. This could be anything from a password to credit card information.
* Our information is only secure while it is being sent over a trusted network, by email or social media application.
* Many persons don’t have a simple screen lock on their phone, removing the first layer of security on a device
* The security of that information is comprised if someone simply takes your unlocked phone or computer (or you give it to them) and opens the email or messaging application you’ve already signed in to.
* A stolen device is the easiest way to get access to someone’s information.

**OBJECTIVE OF SOLUTION**

* Allow users to encode any text message to an unreadable form
* Allow users to create files with encoded messages inside
* Ensure that sensitive information stored on a device locally is protected against prying eyes
* Allow users to send an encrypted message and files to anyone without ever disclosing information.

**TECHNIQUES OF ANALYSIS**

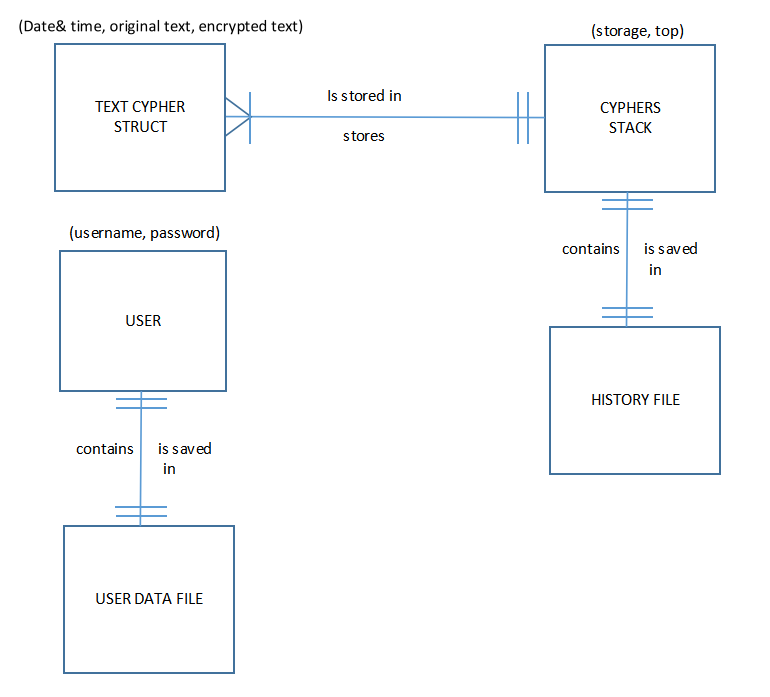
**REQUIREMENTS SPECIFICATION**

The method of data collection used for requirements engineering of ZenCypher was a survey. Surveys are simple to create and easy to distribute using modern website tools. It generally doesn’t take very long to complete a survey and because it is done electronically the data can easily be quantified and made into charts and graphs. Lastly, an online survey encourages honest responses due to the sense of anonymity on the web.

On September 20, 2019, a survey of 25 high school students at Campion College was conducted using Google Forms to gather information about if and how they use encryption in there lives. The survey was distributed via Instagram and Whatsapp messaging. The information gathered was useful in determining how ZenCypher should function.

**SYSTEM MODELS**

**Data Flow Diagram - Context Diagram:**

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Data Flow Diagram -** **Level 0:**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Entity Relationship Diagram:**

B

C

D1 User Data File

**2**

**Accept Menu input**

Successful login

**3**

**Encoding Process**

A

Encrypted text

**4**

**Decoding Process**

D2 History File

Encrypted text

**5**

**Display history**

Decrypted text

Previous encryptions

**6**

**Update user login**

New credentials

D2 History File

User

**1**

**Login**

User credentials

Saved user credentials

D

**FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS**

Functional Requirements:

* The system shall require user sign in by username and password 
* The system shall accept user input as plain English and produce a cyphered version of the text as well as store it in a file 
* All cyphered text shall be saved in a separate file for later recollection
* The system shall allow users to input a custom filename for saving
* The system shall allow the user to decoder cyphered text and files
* The program shall implement a command line interface

Non-Functional Requirements:

* The program will provide an easy to use interface
* The program will allow more than one user accounts
* The system will allow the user to modify their login details at any point
* The program should be written in the C programming language
* The program should be developed using the CodeBlocks IDE

**DESIGN SPECIFICATION**

**System Architecture (HIPO chart):**

1.0

ZenCypher Encryption System

2.0

Login or Sign Up

3.0

Main Menu

4.0

Text Encoding

8.0

Exit

7.0

Change login info

6.0

View History

5.0

Text Decoding

6.1

Clear History

4.1

Save To File

**Annotations:**

1.0 - ZenCypher encryption system;The program starts with a screen showing the title and version then calls the login function

2.0 - Prompts the user for their username and password if they have an account created, otherwise it prompts the user to create one

3.0 - The main screen of the program; The user is presented with options for using the program.

3.1 - The encoding process; the user is prompted for a string of text and the program converts it to cyphered text

3.1.1 - After encoding a string the resulting cypher is saved to a file for decoding later

3.2 - The decoding process; the user is prompted for a cyphered text or file and the program converts it back to readable English text

3.3 - All previous encryptions are displayed to the user showing the original text and the cyphered version as well as the date and time it was done

3.4 - The user is prompted for a new username and password and the user data file is updated

3.5 - A goodbye message is displayed and the program self closes

**RECORDS:**

**TextCypher** Record 🡪 Contains information about an encoding

Record TextCypher

original: String

encoded: String

dateTime: String

EndTextCypher

**User** Record 🡪 Stores the current logged in user

Record User

username: String

password: String

EndUser

**DATA STRUCTURES:**

Stack stackOfEncodings

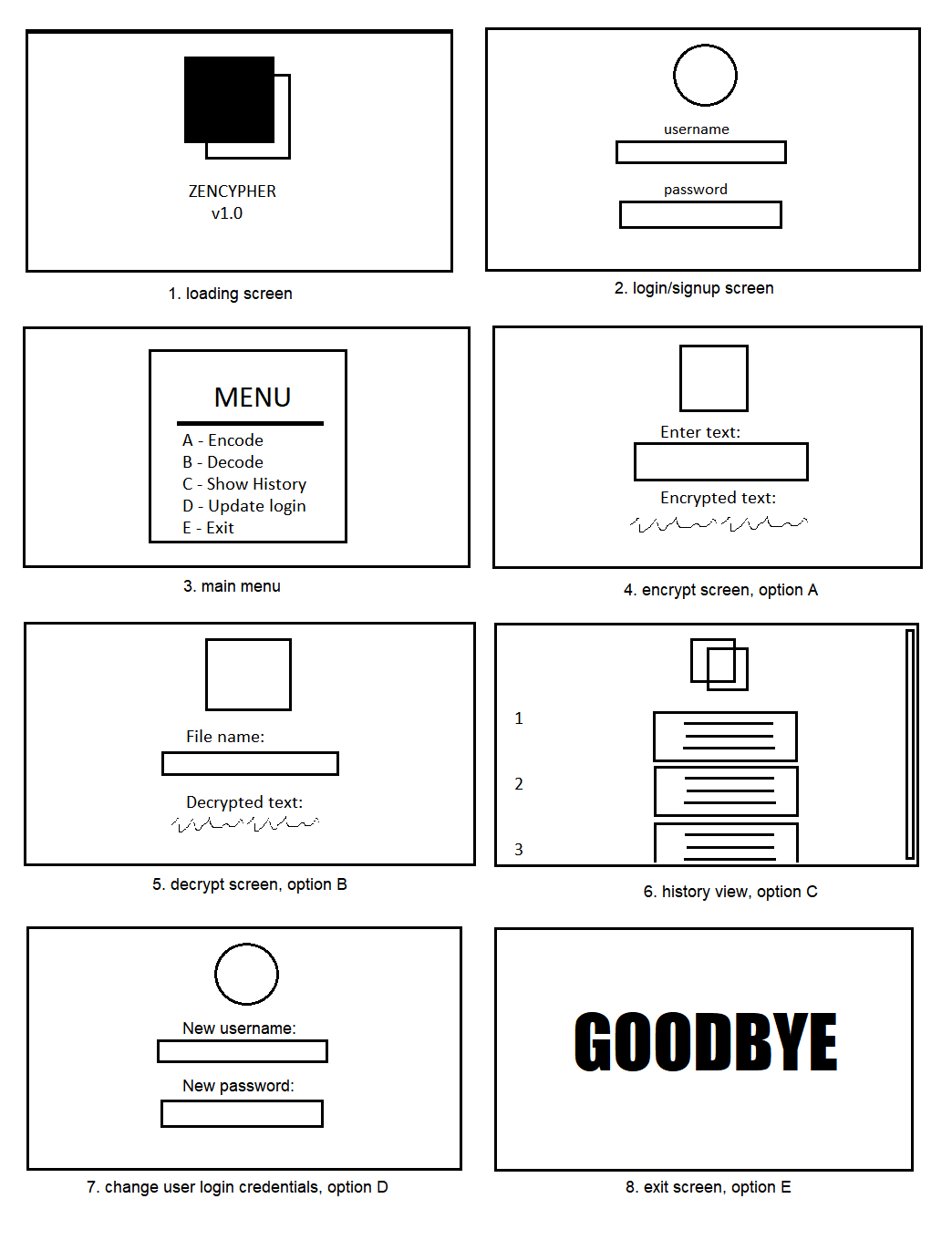
Array storage[]: TextCypher

top: Integer

EndStack

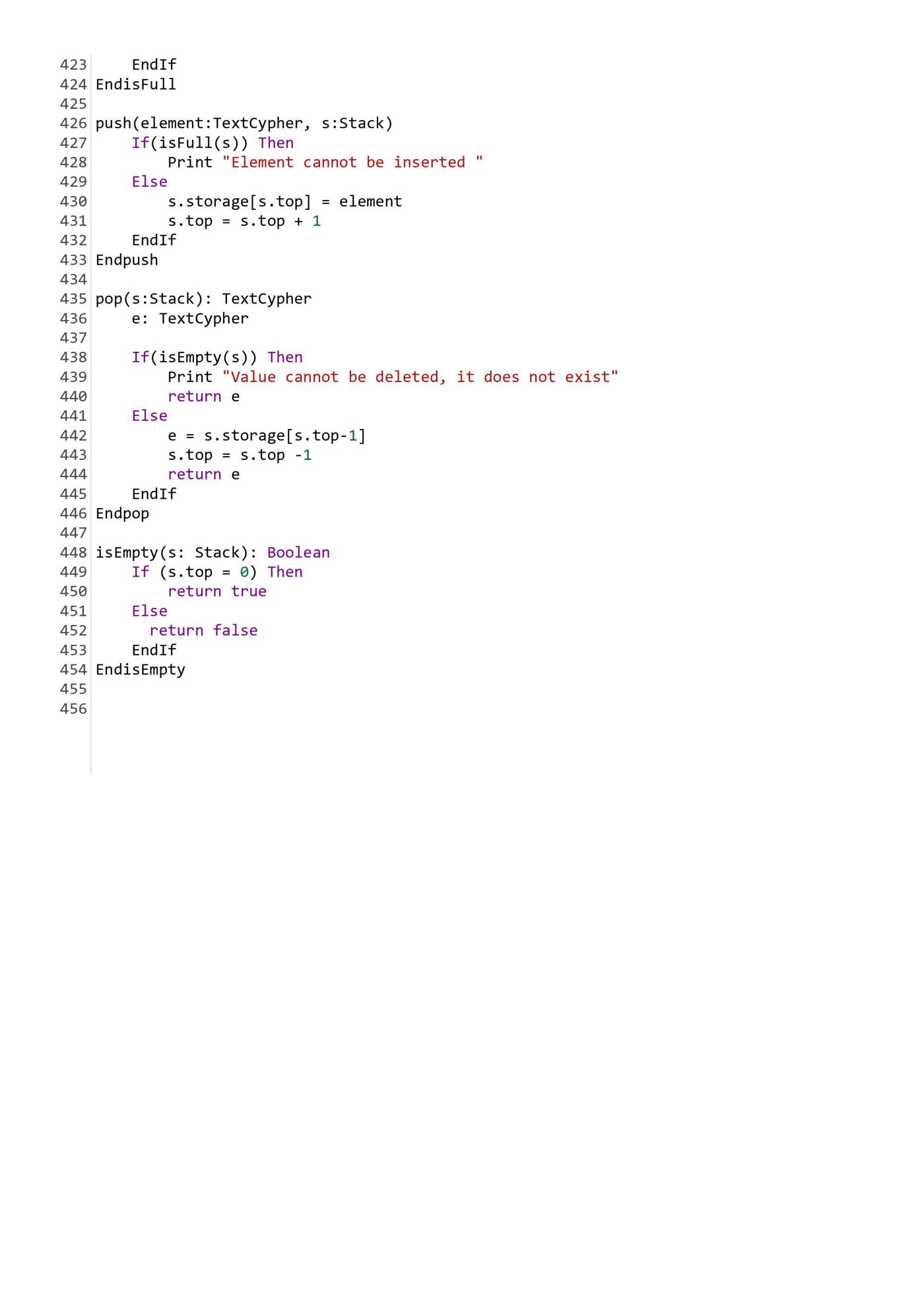
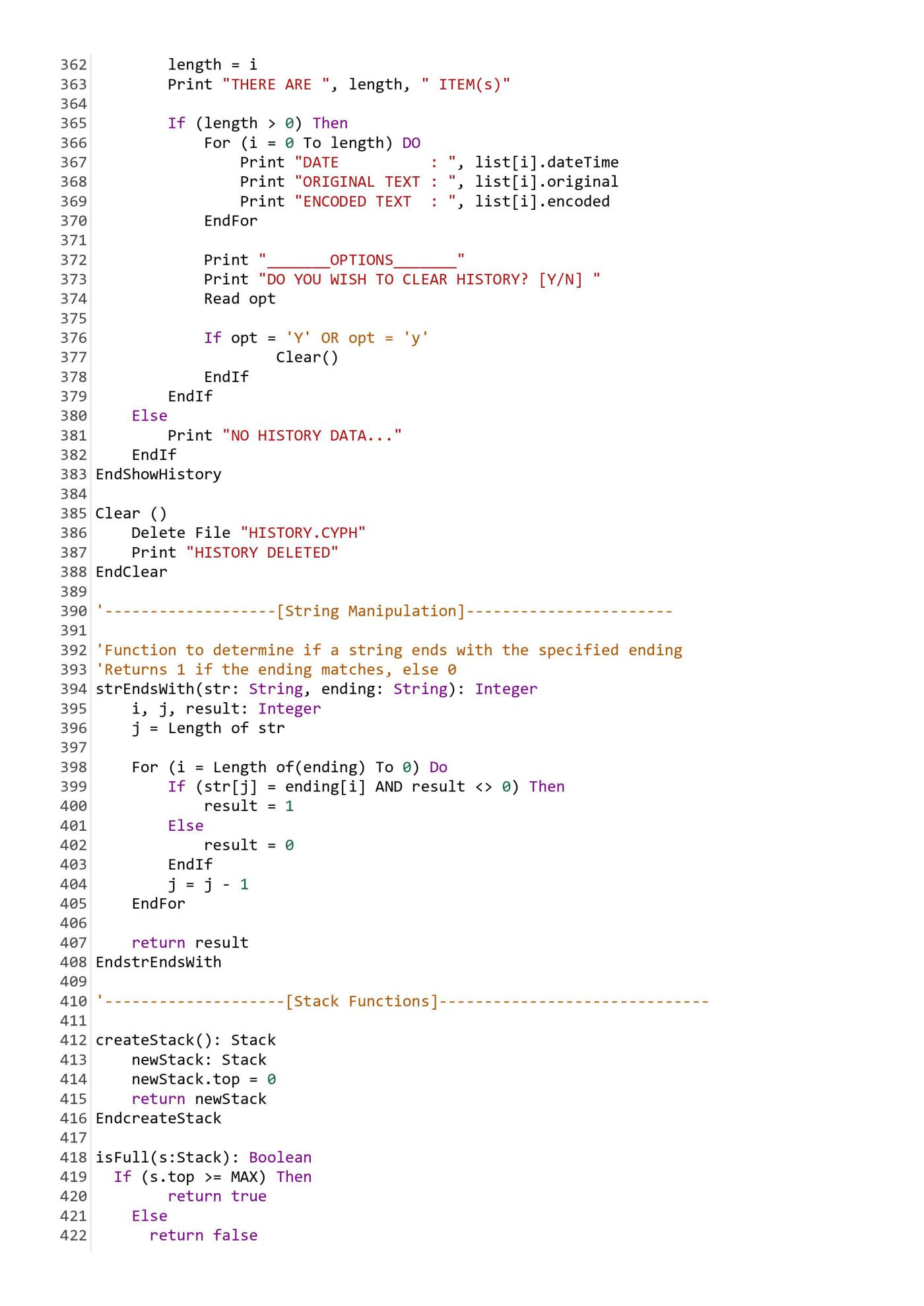
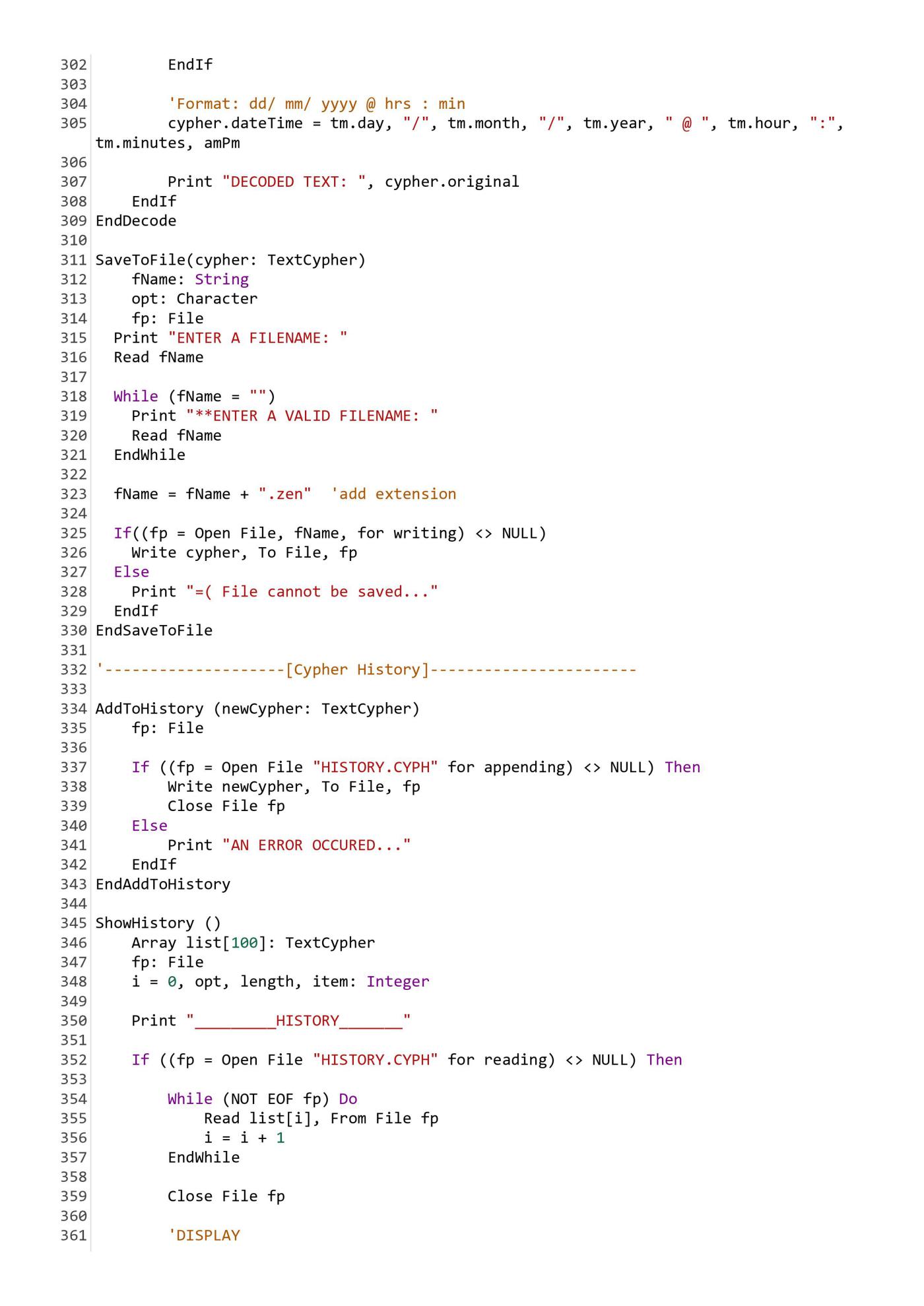
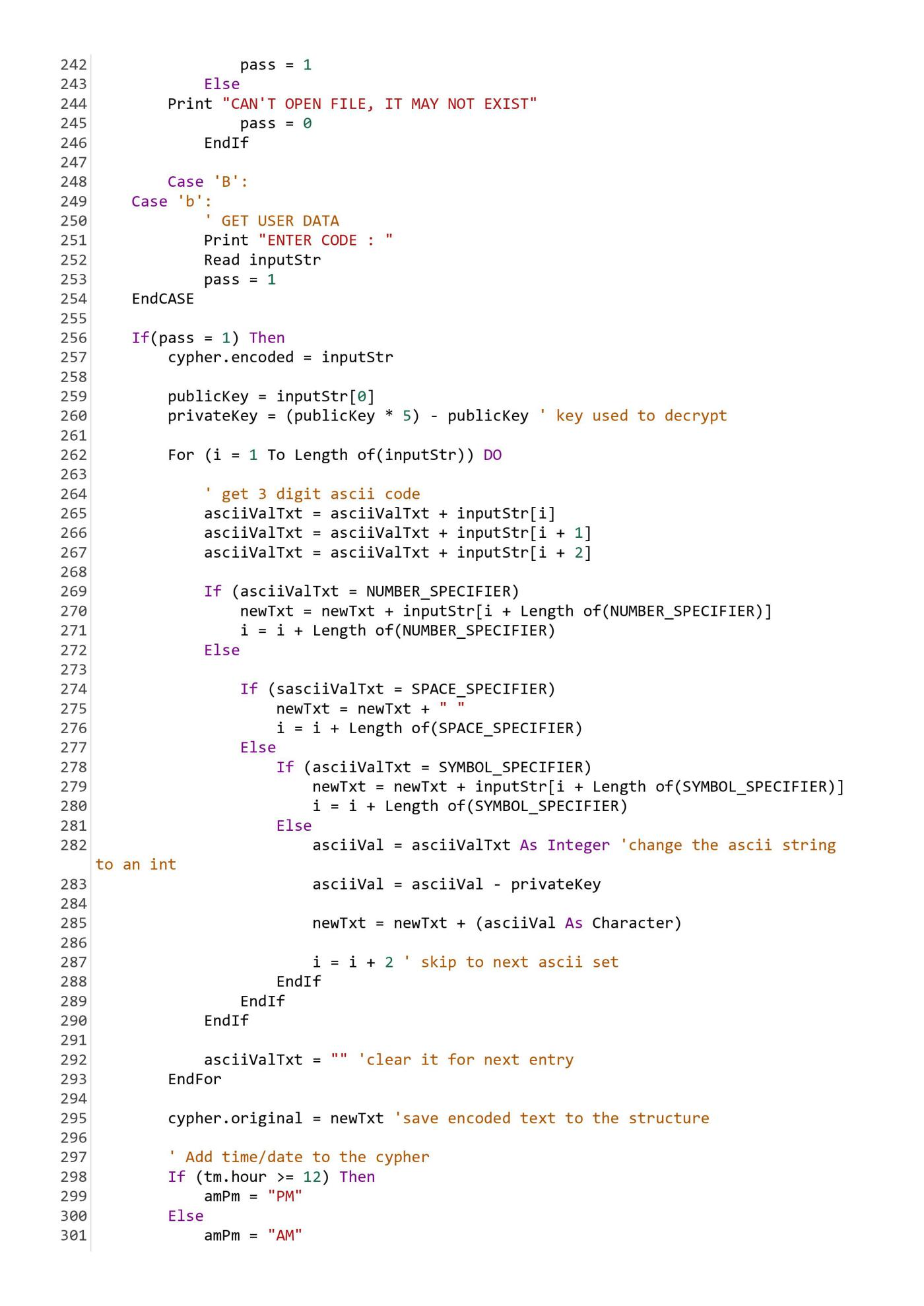
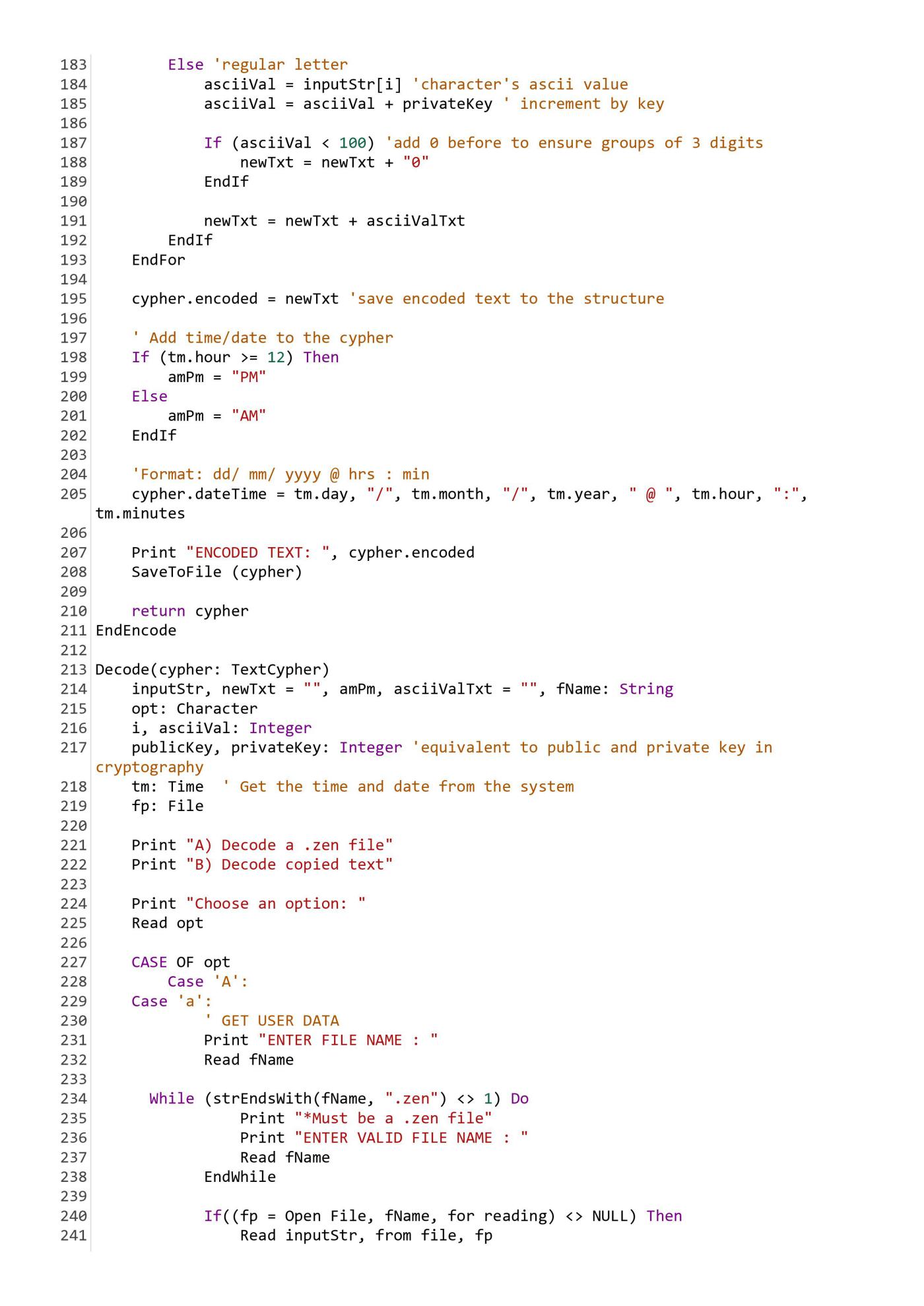
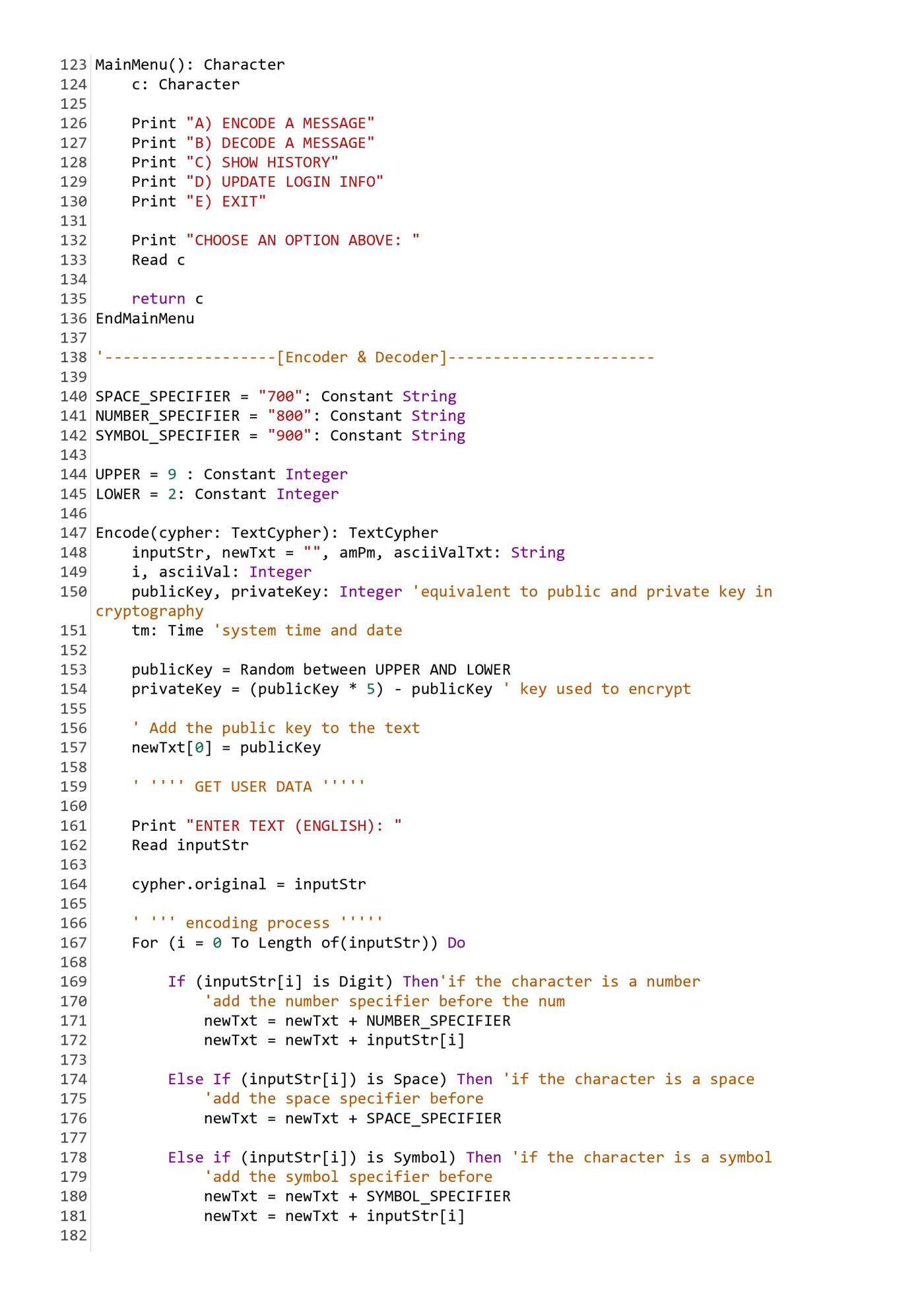
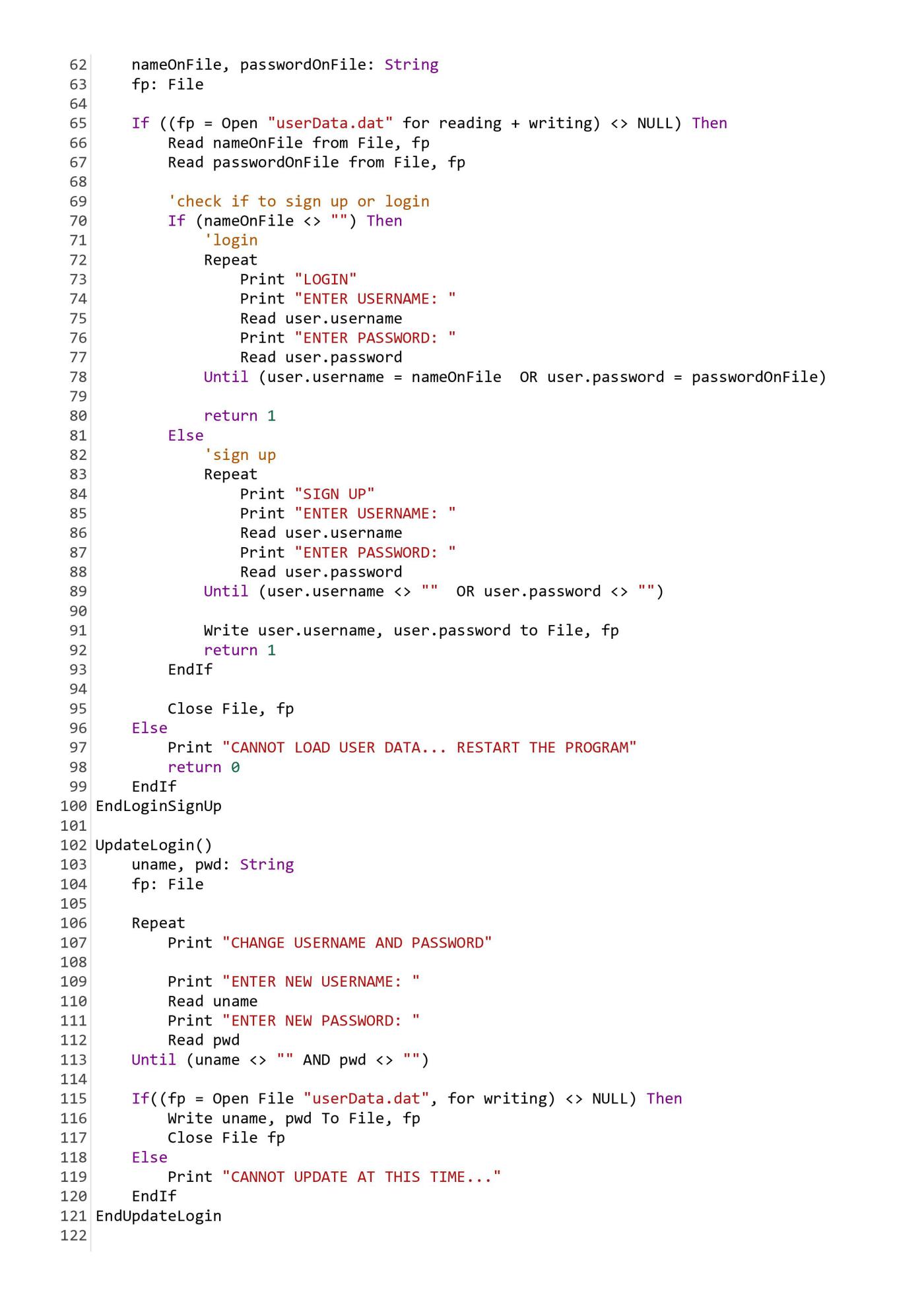
**A Stack** was chosen to store previous encryptions because stacks store data in a LIFO (first in, last out) manner. This means that the last thing the user encrypts will be the first displayed in the history. This is so the user can see their most recent activity.

**Interface Specification:**

A command line interface was chosen because it is extremely fast because it does not use as much processing power as say, a graphical user interface (GUI).

**ALGORITHM/ PSEUDOCODE:**





**Code Test Plans:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Screen** | | **Input** | | | **Purpose of test** | **Output** |
| Sign Up | | Username: “admin”  Password: “admin” | | | To create a user for the program | User is logged in and main menu appears |
| Username: none  Password: none | | | Sign up failed, the user is prompted to re-enter |
| **Screen** | | **Input** | | | **Purpose of test** | **Output** |
| Login | | Username: “admin”  Password: “admin” | | | To login the previously created user to the program | User is logged in and main menu appears |
| Username: “help”  Password: “admin” | | | Login failed, the user is prompted to re-enter |
| **Screen** | **Input** | | **Purpose of test** | **Output** | | |
| Main Menu | ‘A’ or ‘a’ | | To check that the correct menu options are executed on keypress | Encode function is called | | |
| ‘B’ or ‘b’ | | Decode function is called | | |
| ‘C’ or ‘c’ | | ShowHistory function is called | | |
| ‘D’ or ‘d’ | | UpdateLogin function is called | | |
| ‘E’ or ‘e’ | | Goodbye function is called.  Program ends | | |
| Any other character | | INVALID OPTION.  Prompts user to try again | | |
| **Screen** | **Input** | | **Purpose of test** | **Output** | | |
| Encode | Text: “Hello World”  Filename:  “Hello” | | To check that the encoding process works | 6096125132132135700111135138132124 | | |
| Encoded text saved to file Hello.zen | | |
| **Screen** | **Input** | | **Purpose of test** | **Output** | | |
| Encode | Text: “Hello World”  Filename:  “Hello” | | To check that the encoding process works | 6096125132132135700111135138132124 | | |
| Encoded text saved to file Hello.zen | | |

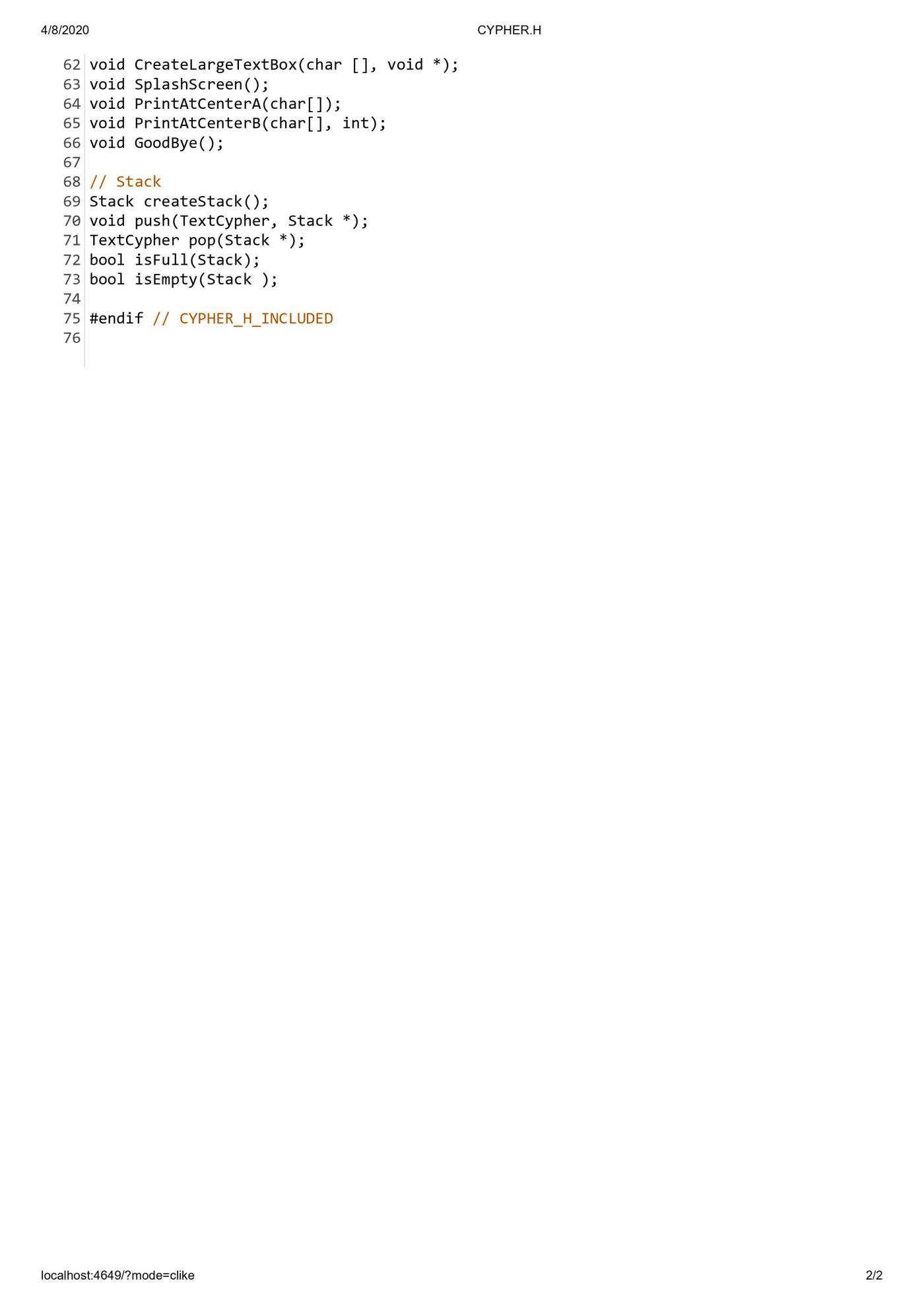
|  |  |  |  |
| --- | --- | --- | --- |
| **Screen** | **Input** | **Purpose of test** | **Output** |
| UpdateLogin | Username: admin  Password: pwd123 | To check that the user is able to successfully change his/her username and password | Updates the user’s login information. Returns to main menu |
| Username: none  Password: none | Update fails, user is prompted for valid data |

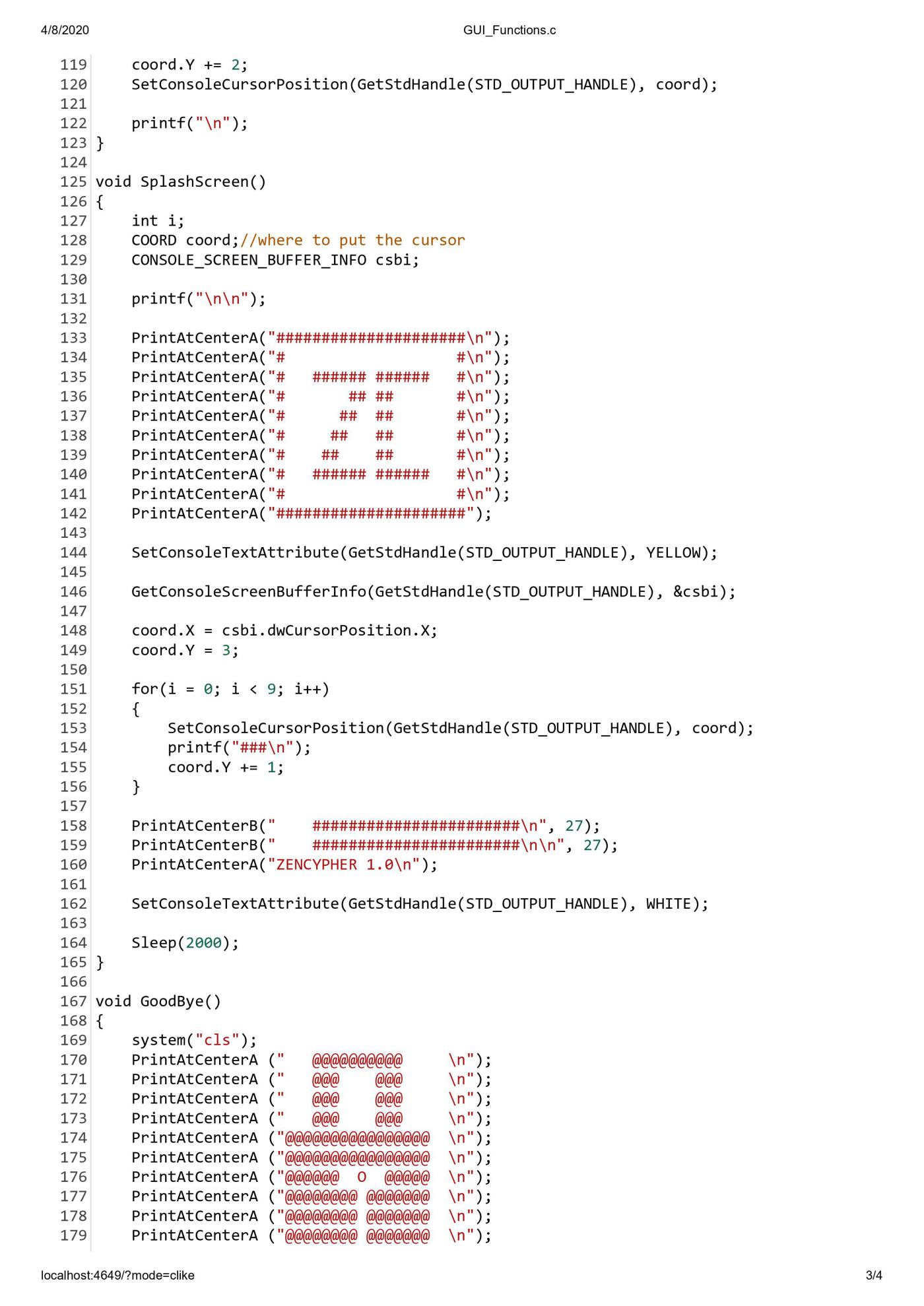
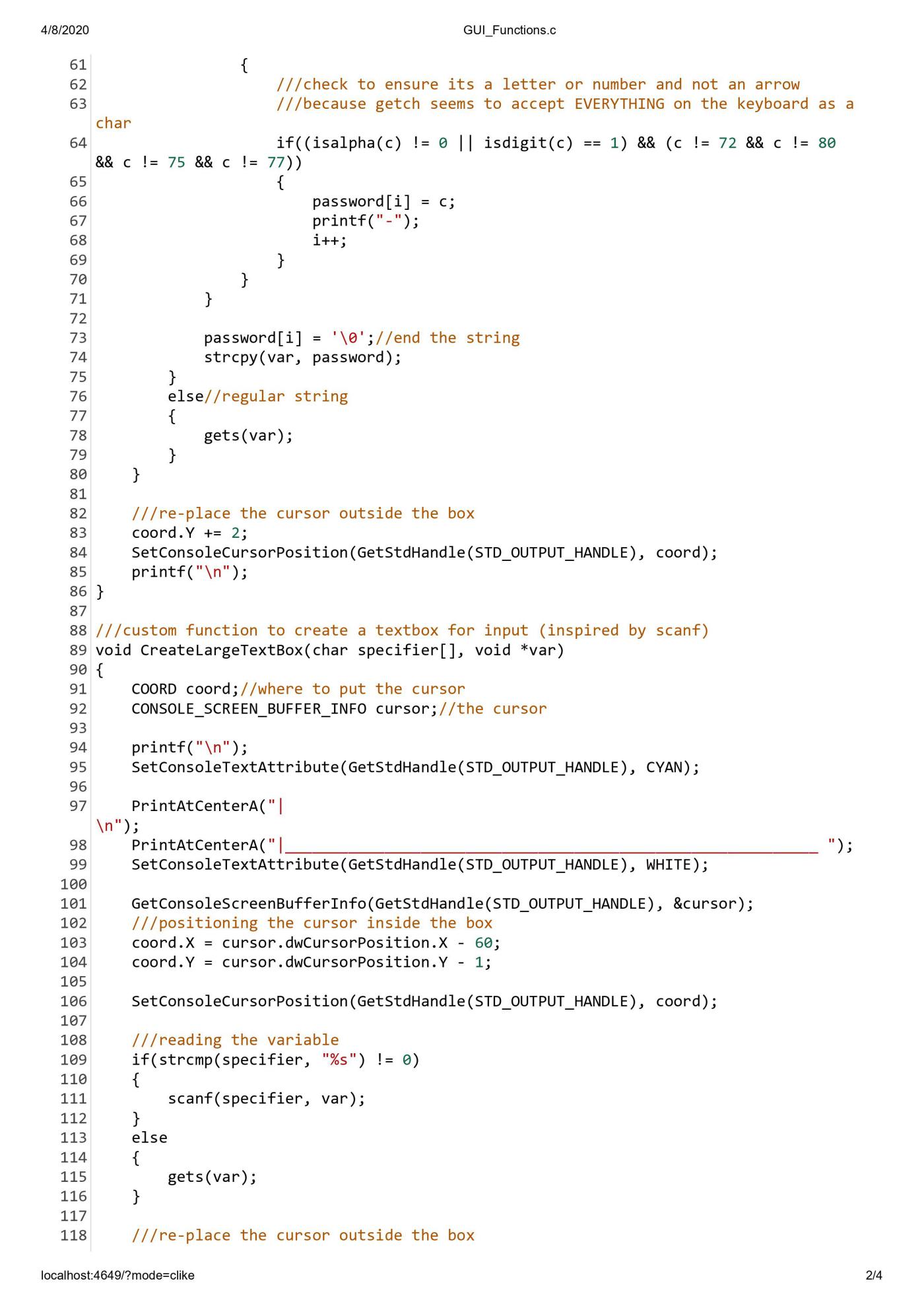
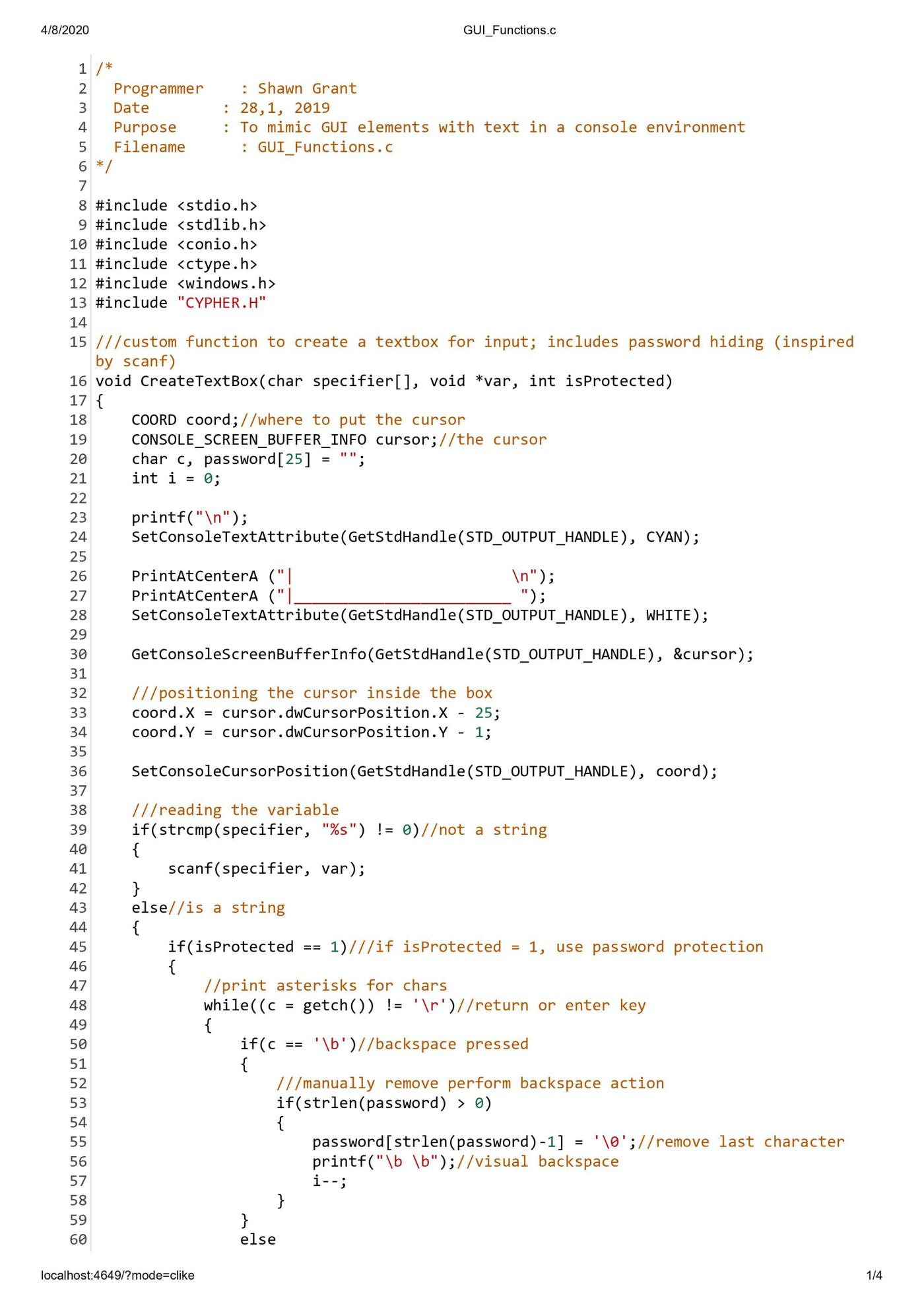
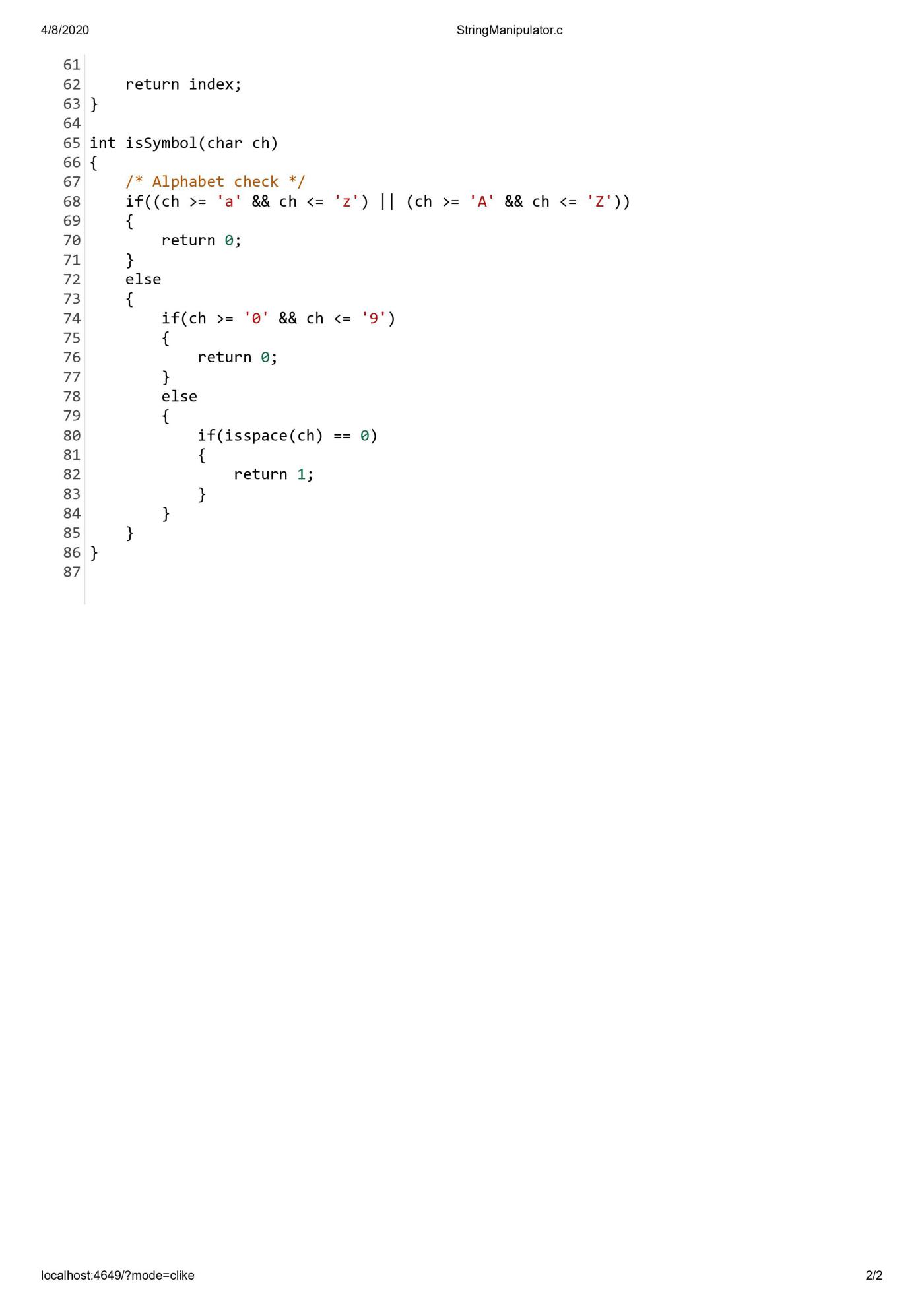
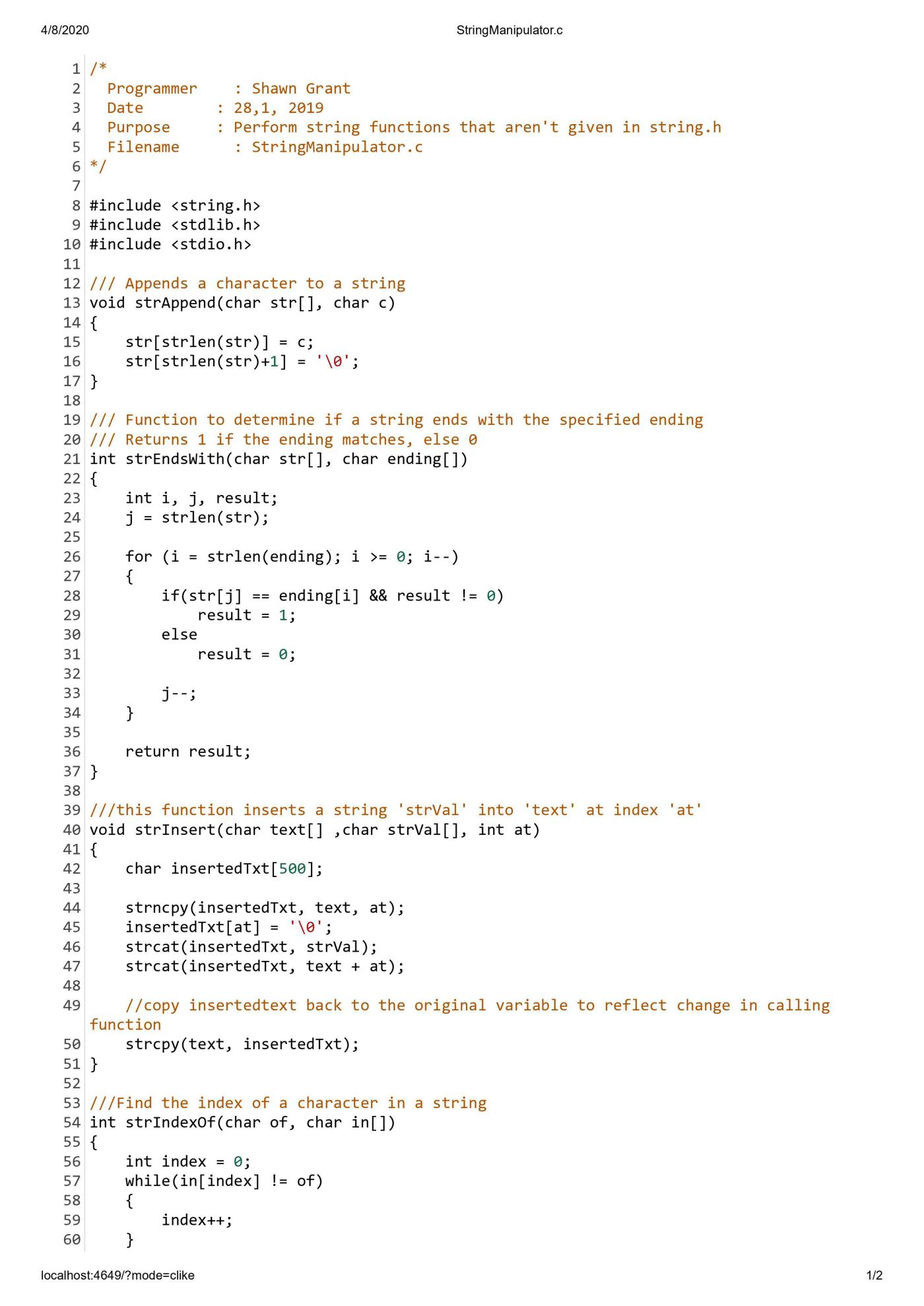
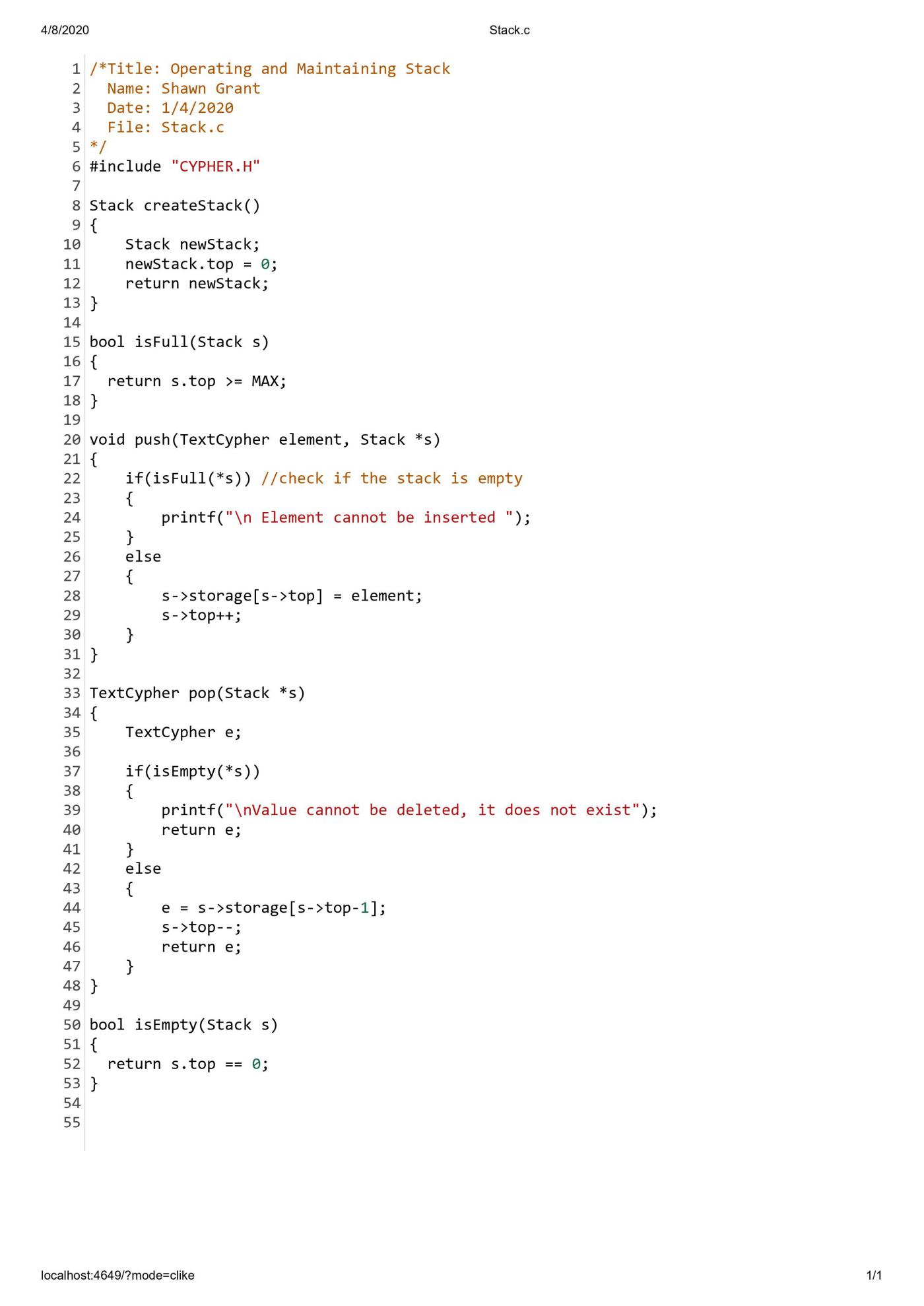
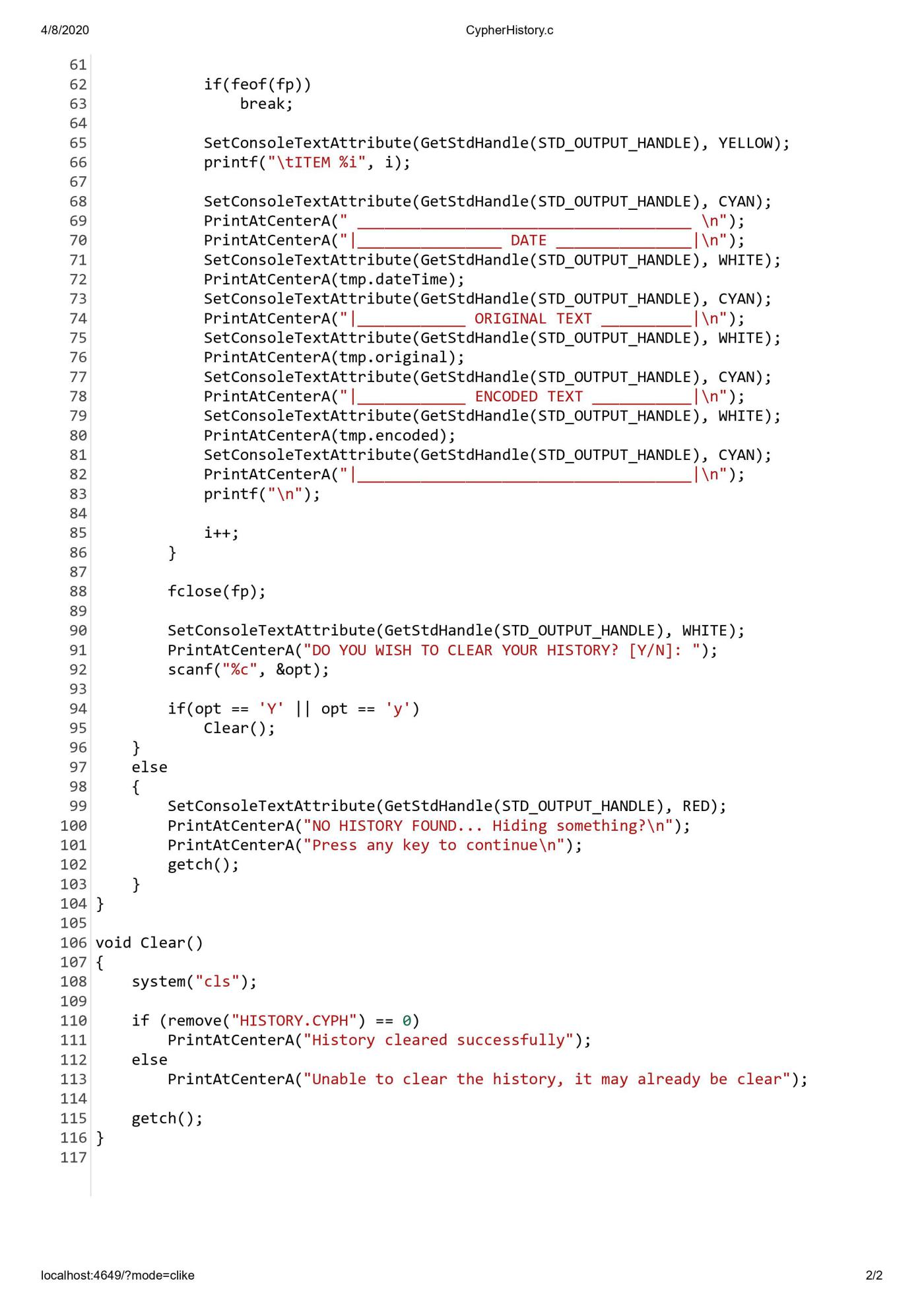
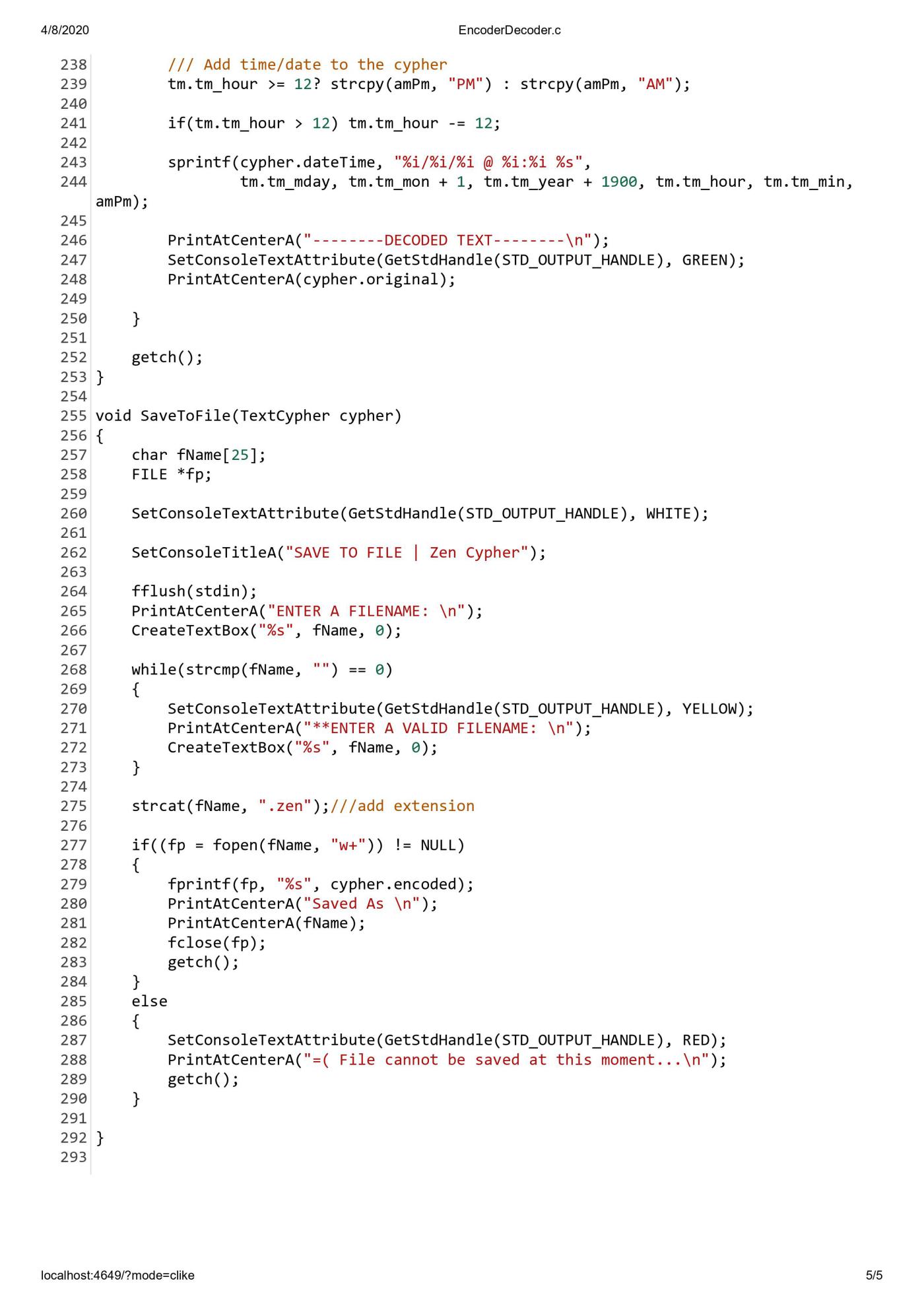
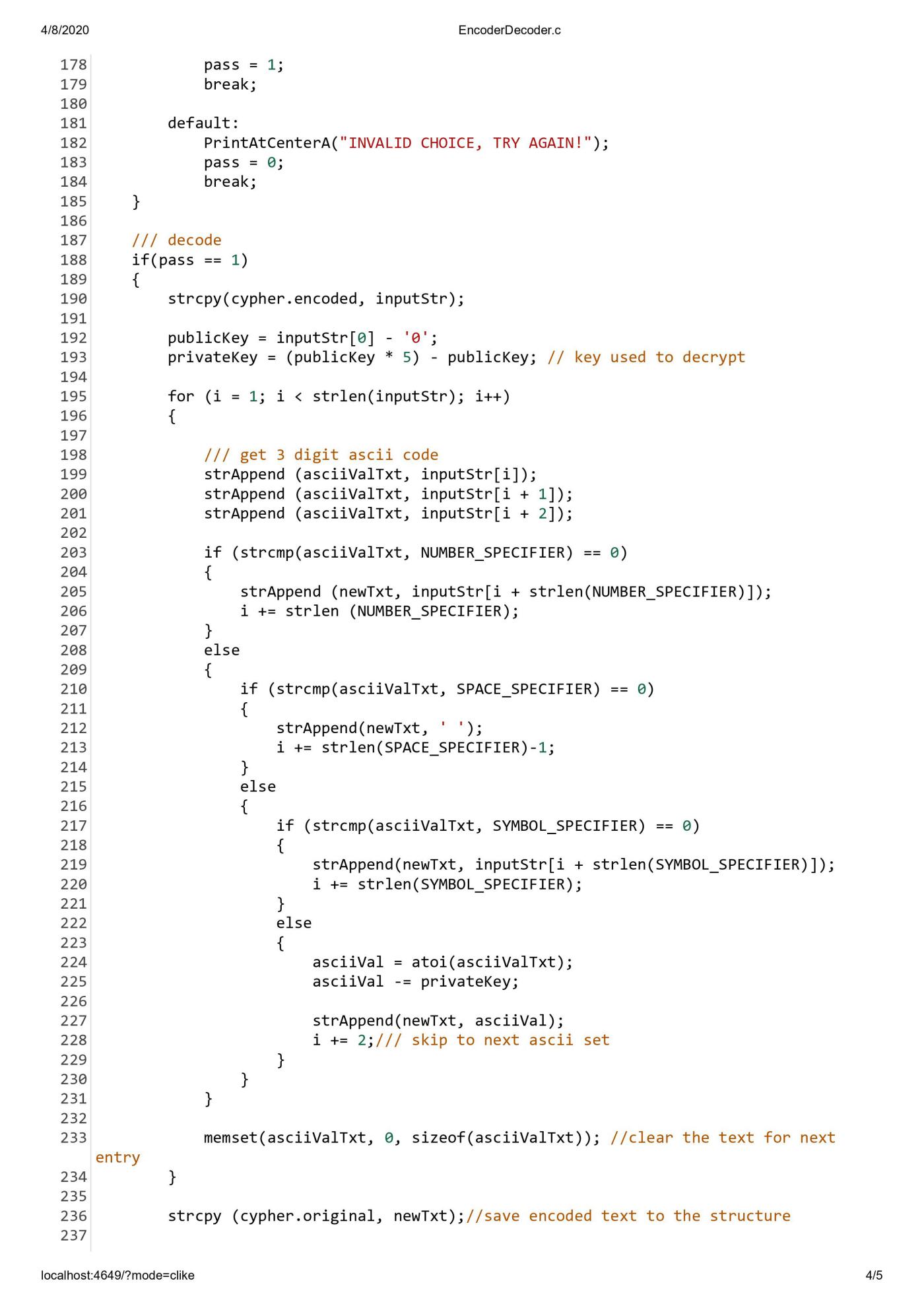
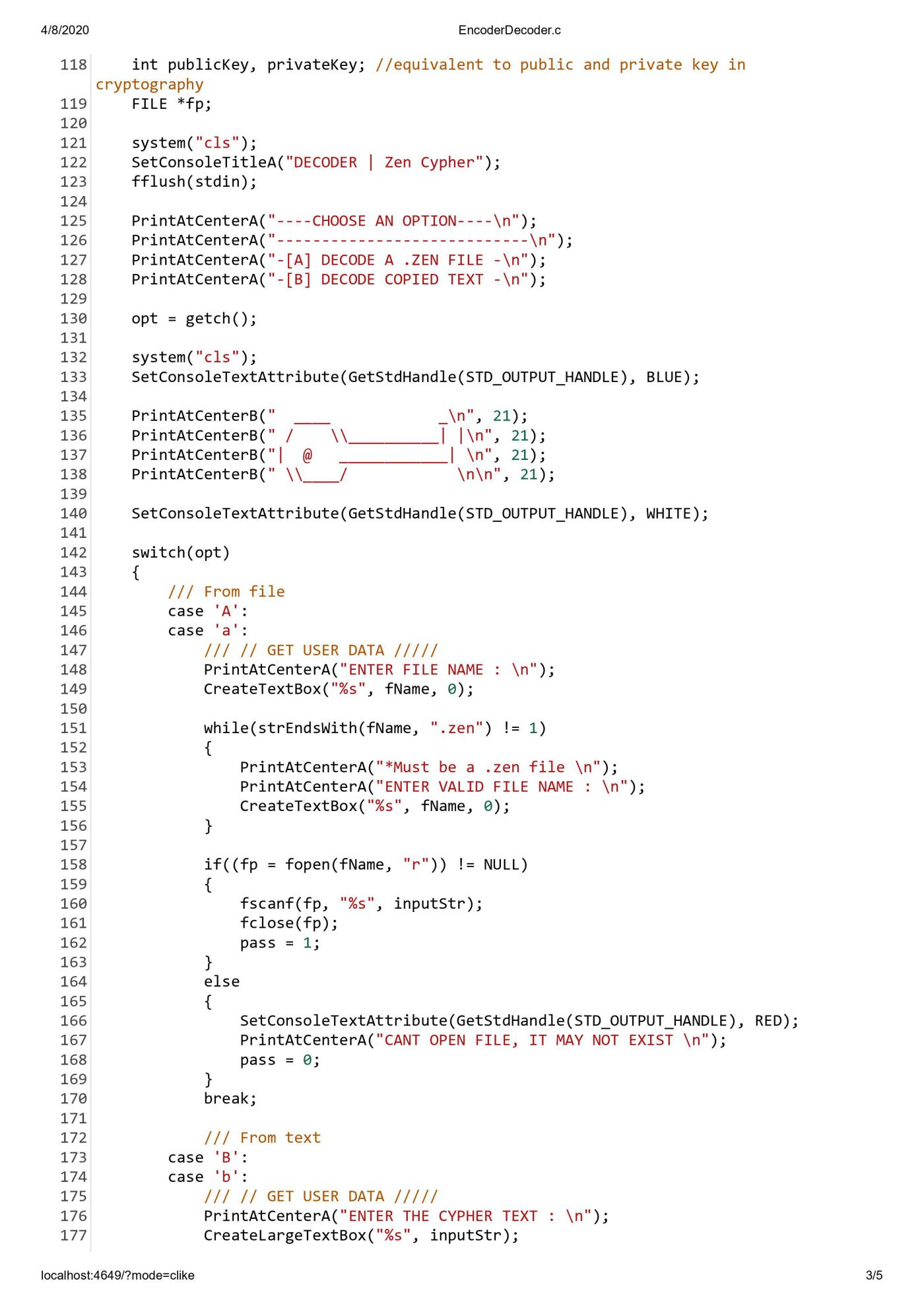
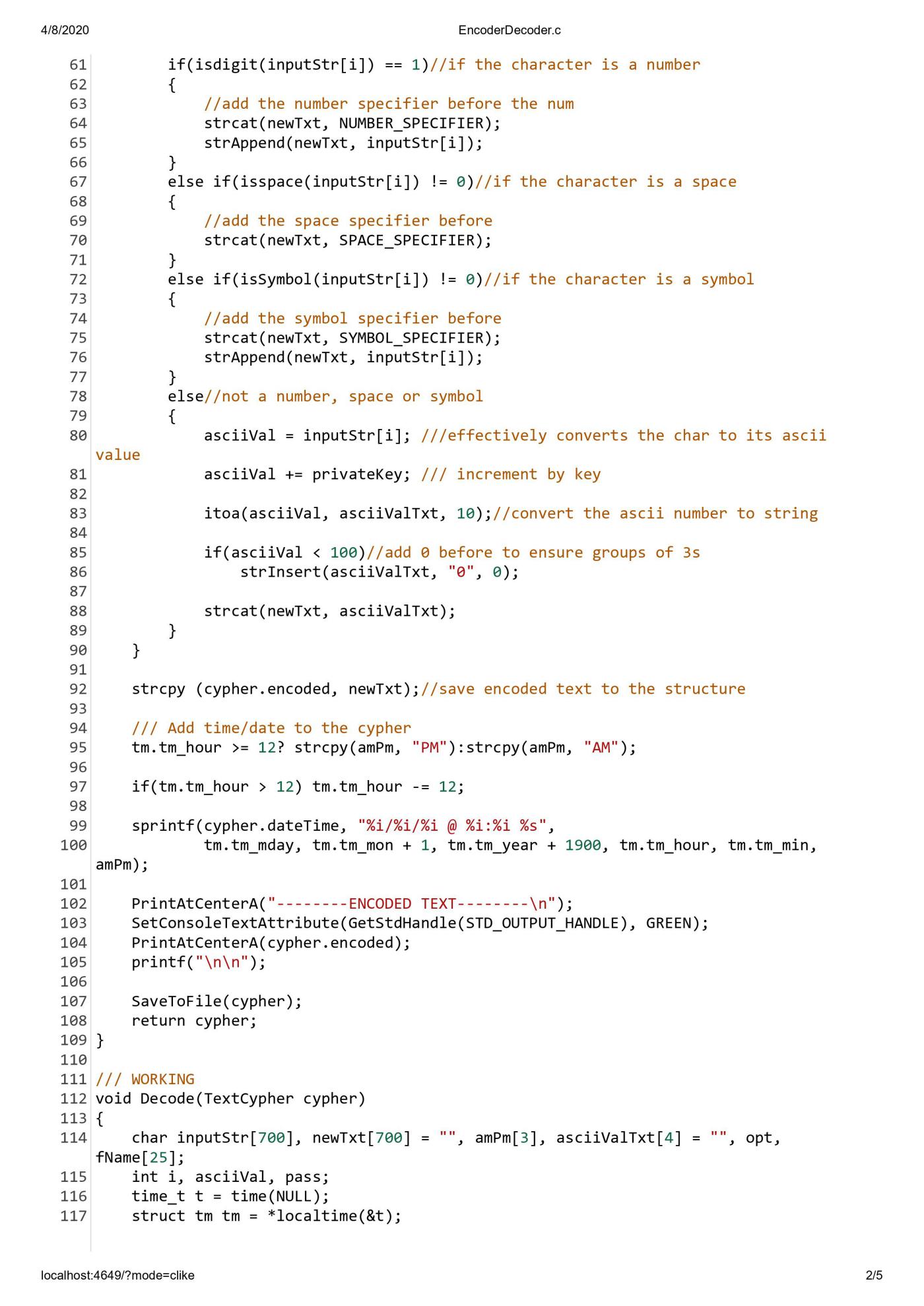
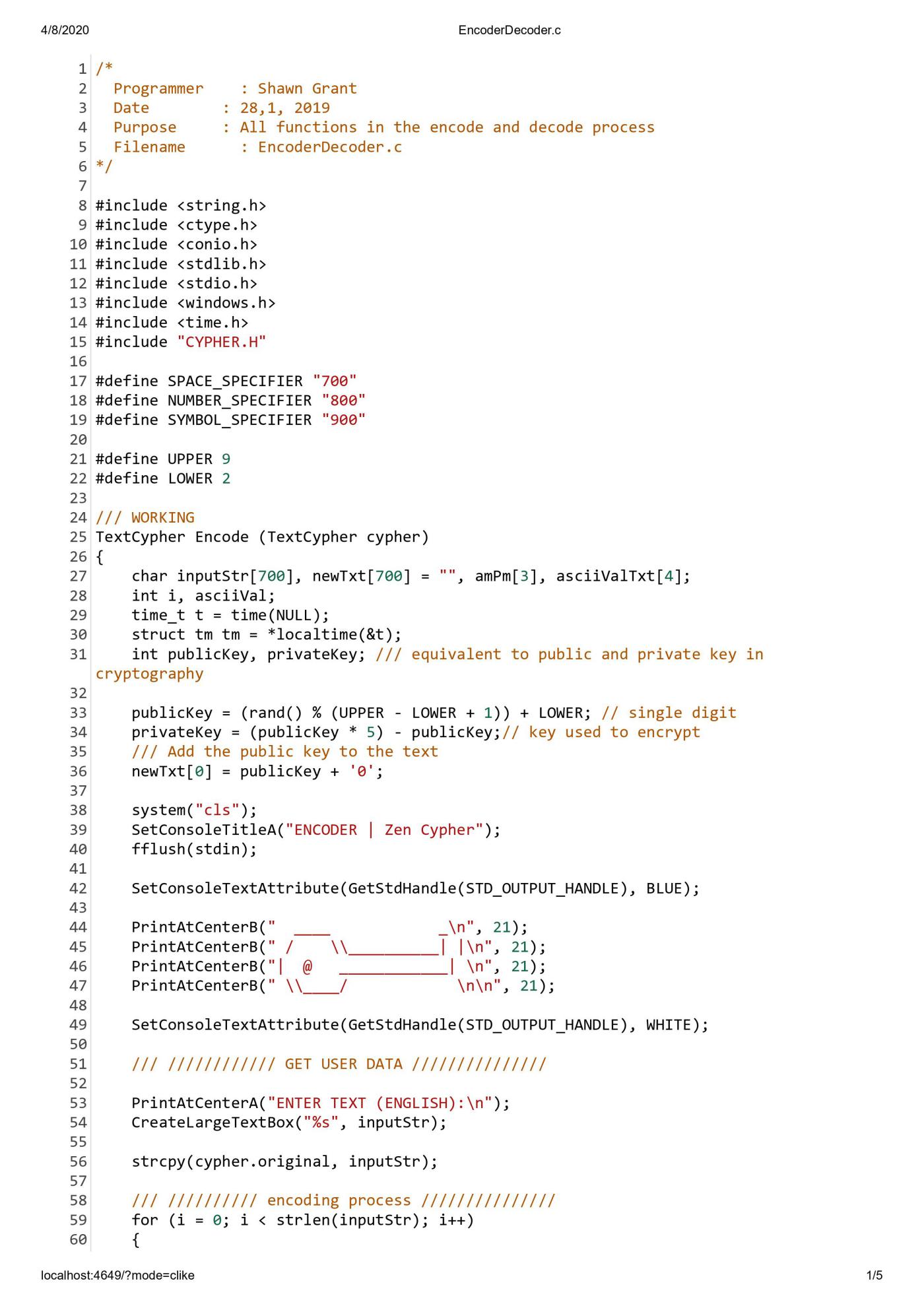
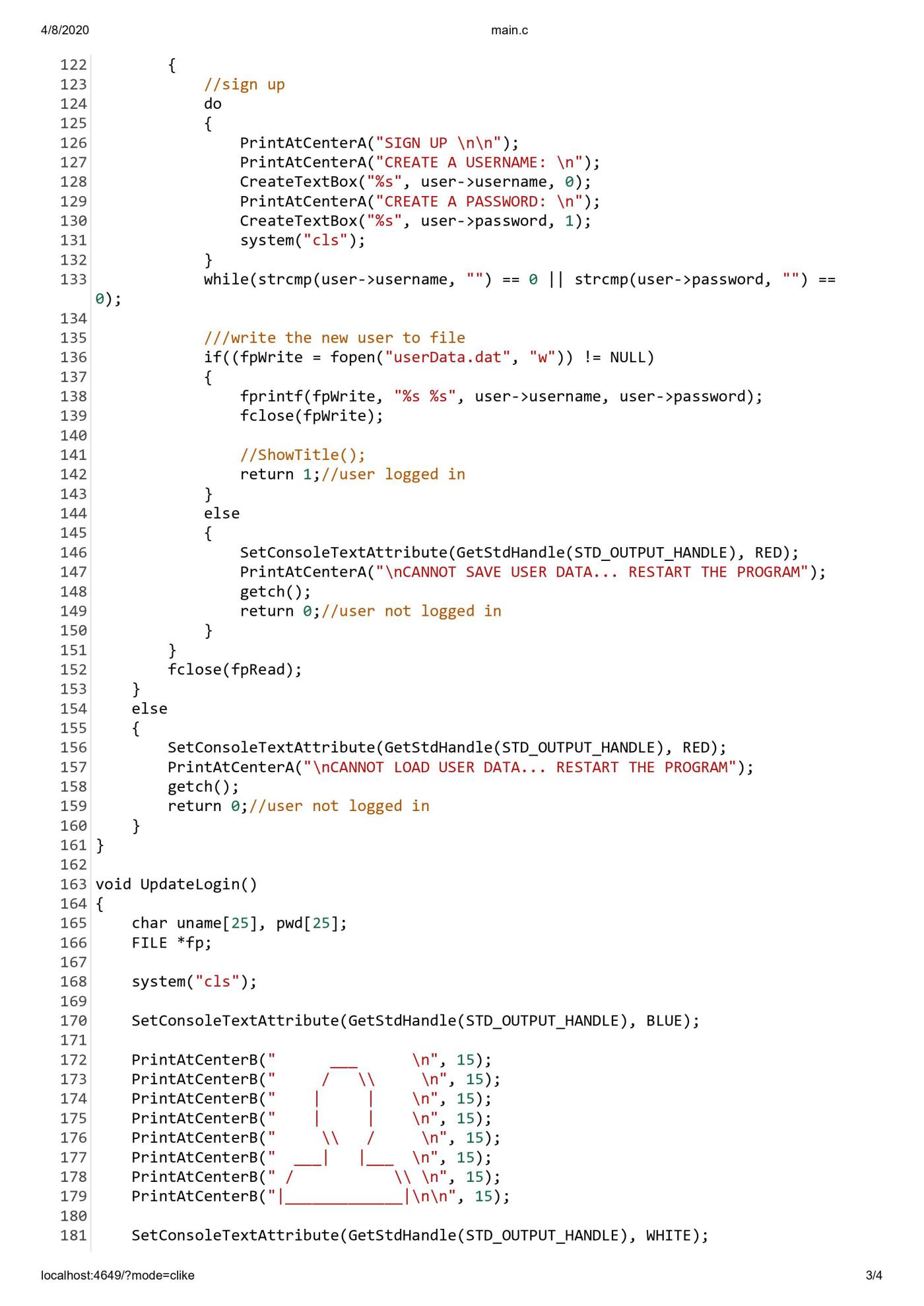
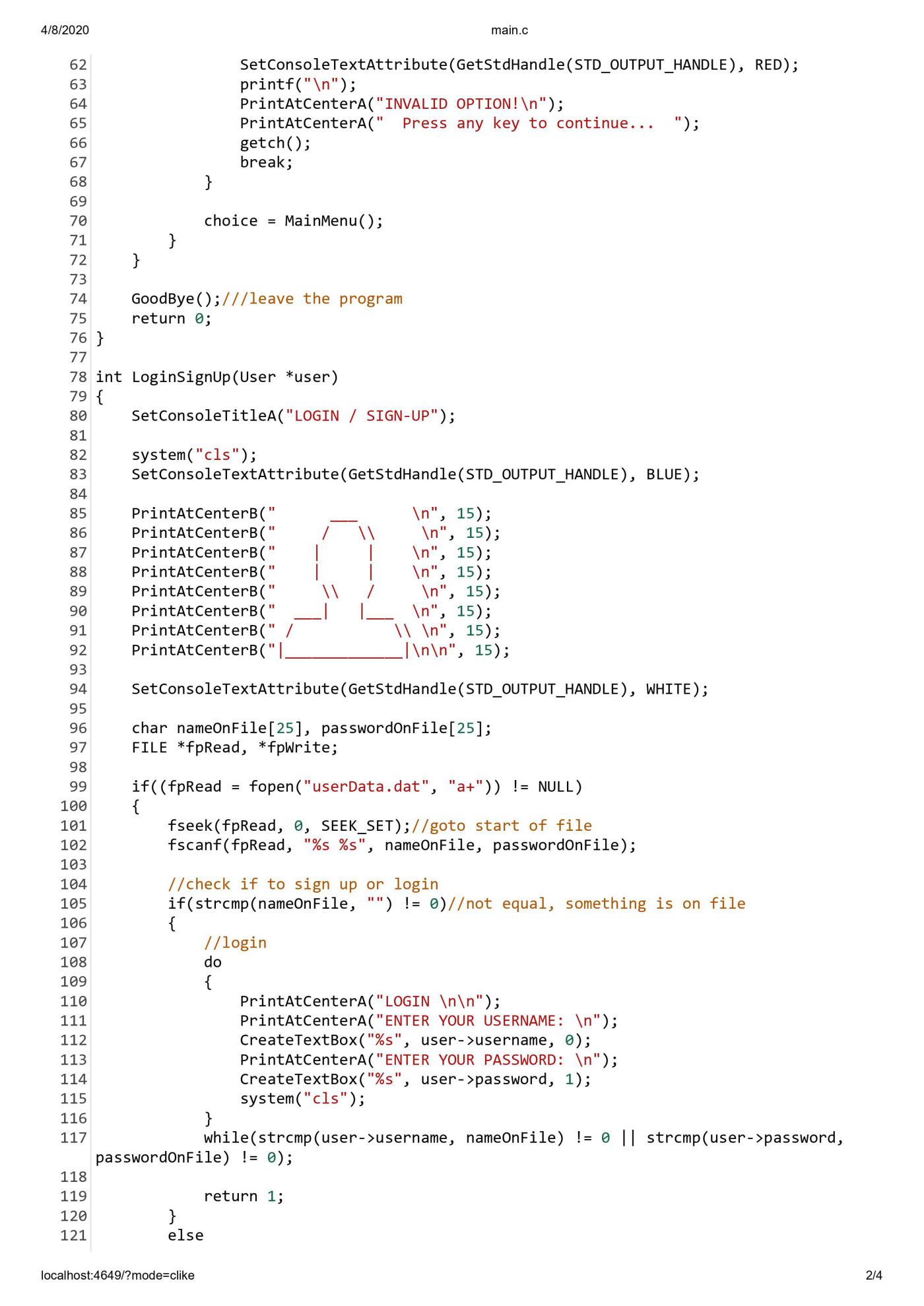
**APPLICATION DEVELOPMENT**

**C Code:**



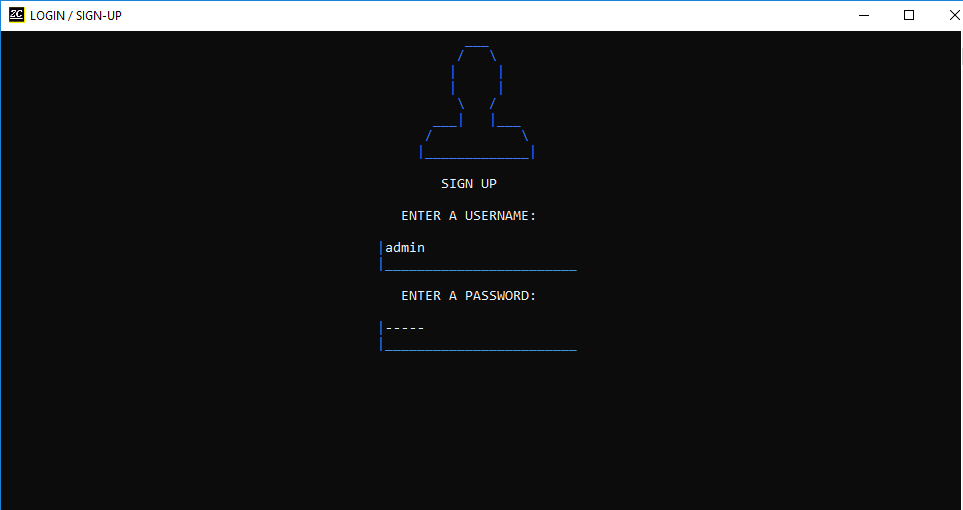






**Code Test Plan Results:**

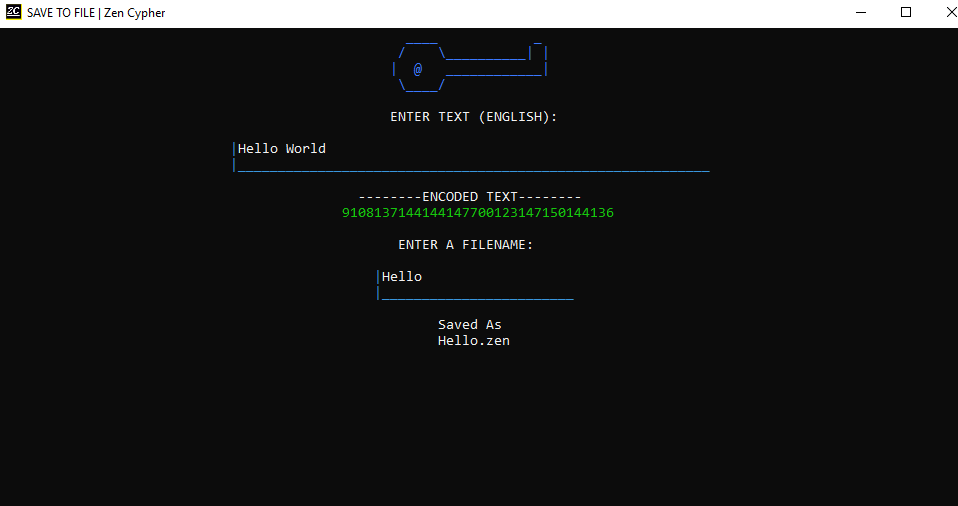
**SignUp:**

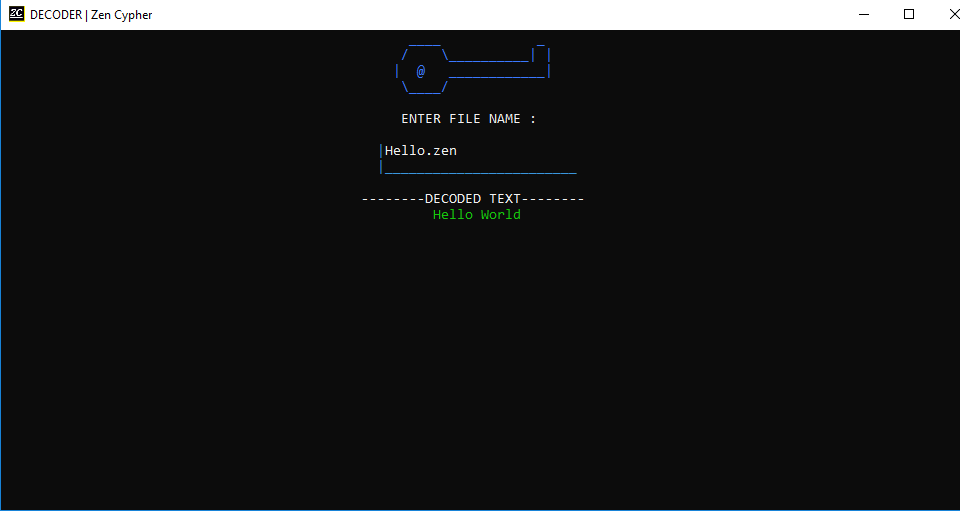


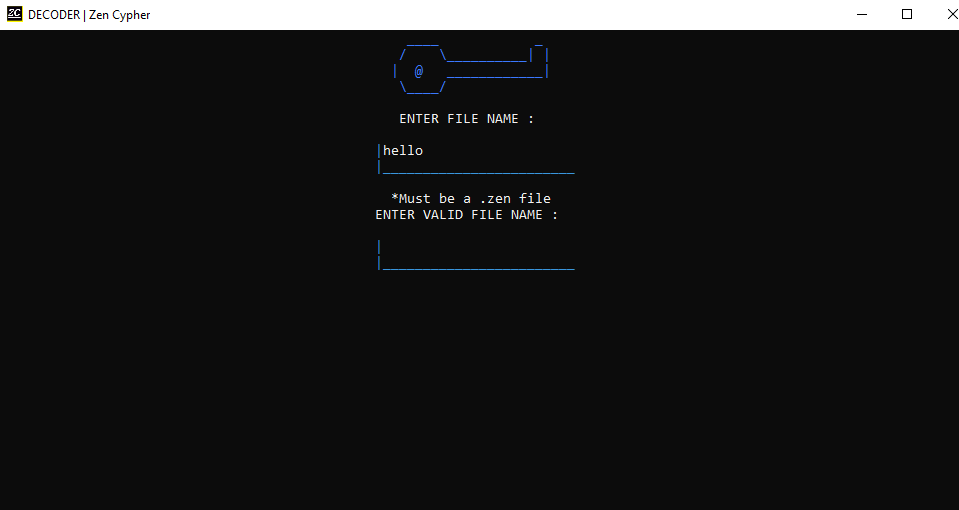
**Main Menu:**

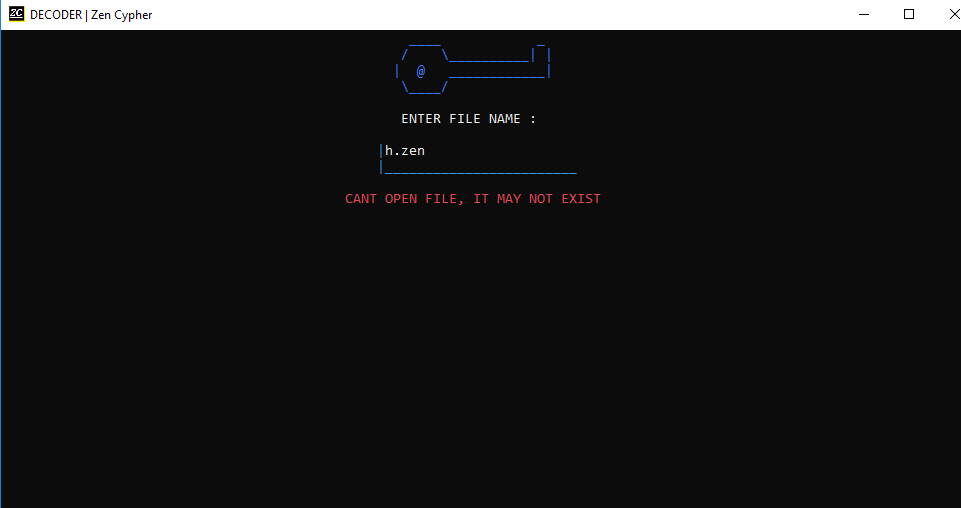


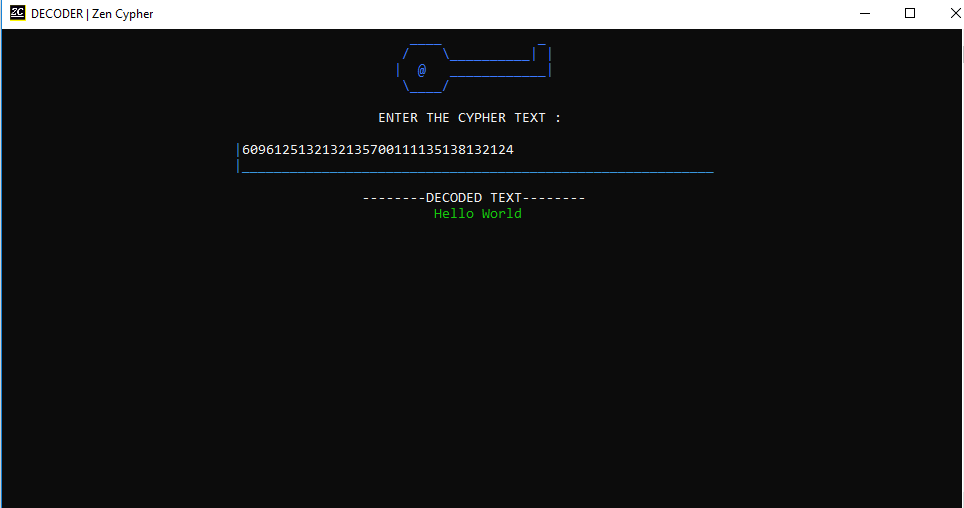
**Encode:**

****

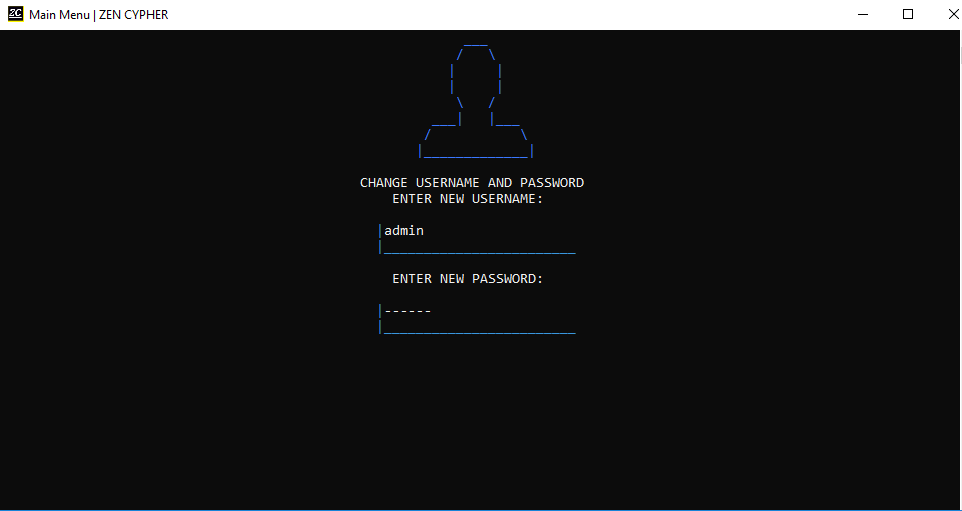
**Decode:**

****

****

****

**UpdateLogin:**

****

**DOCUMENTATION**

**CONCLUSION**

The ZenCypher program had many issues during development leading to delay in completion and certain features being excluded from the final build. The original encryption method had to be abandoned due to limitations of the C programming language in favour of an RSA-like system using public and private keys.

The program’s user interface (UI) was very exciting to build and provided a constant motivational challenge to think outside the box, in order to create GUI-like elements inside a command-line application. Originally it was intended for the user to have the ability to use the mouse to select items on screen, but this feature had to be discarded because of its complexity.

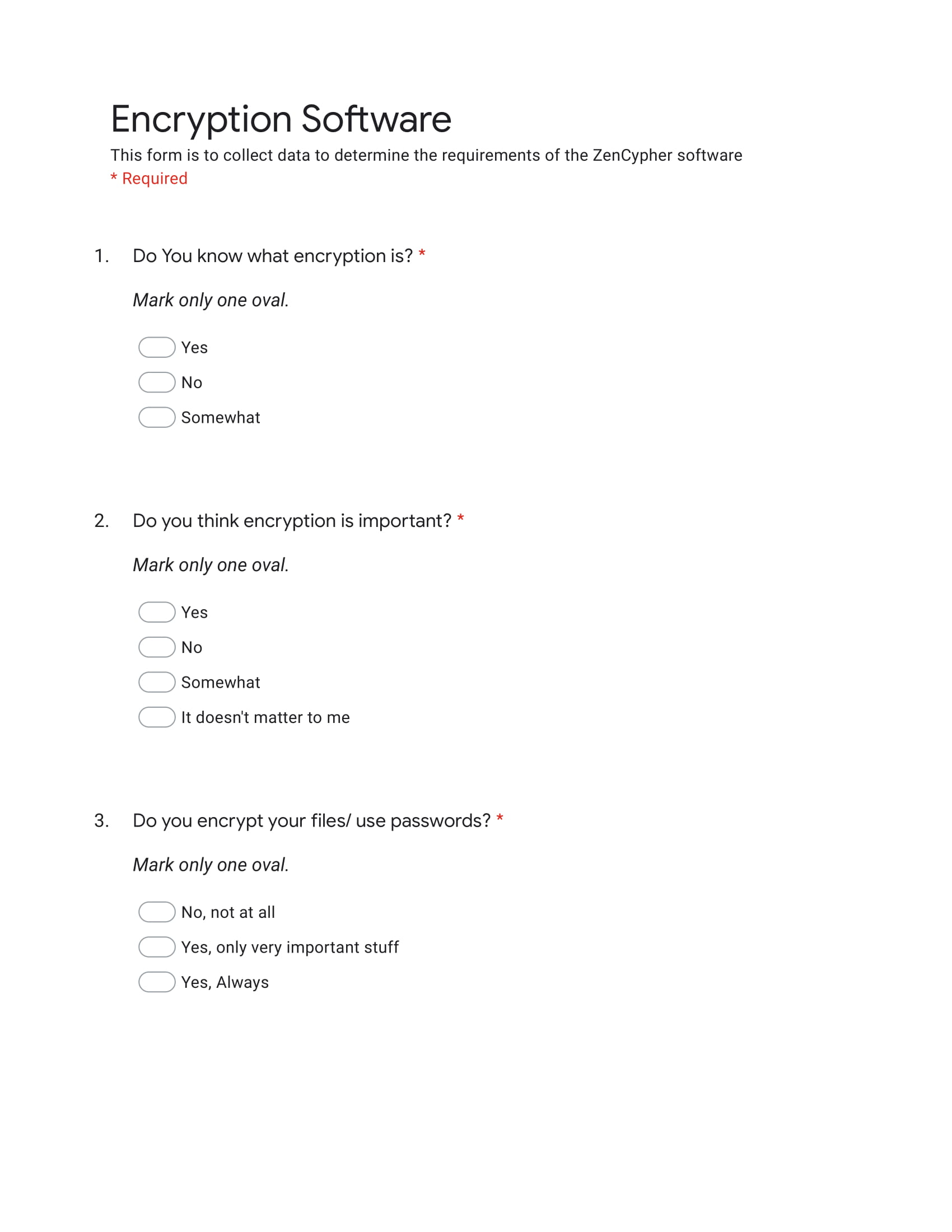
A loading animation was also to be implemented but was removed because it was not consistent with the subtle UI of the rest of the program. A menu navigable by arrow keys was also created but could not be finished in time and had to be removed

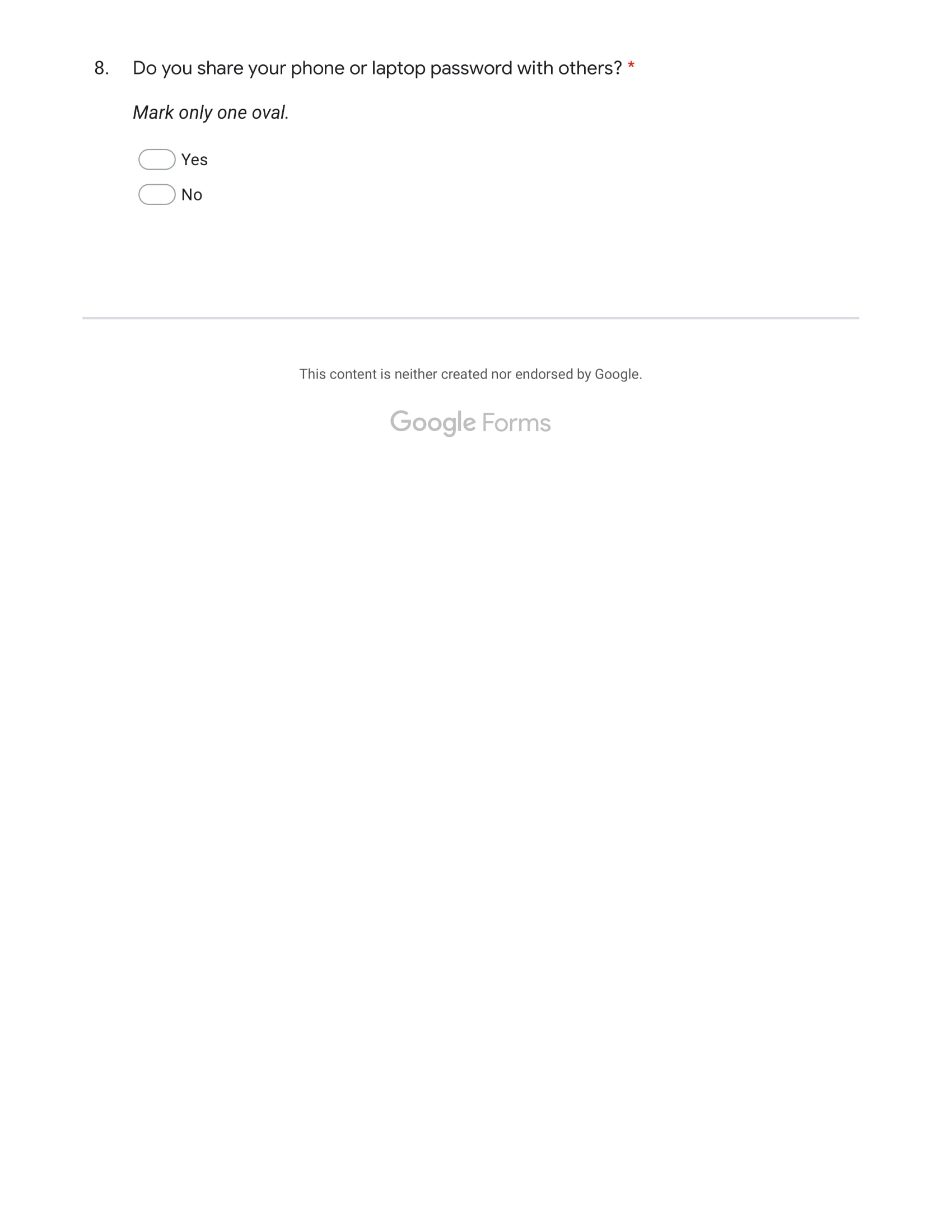
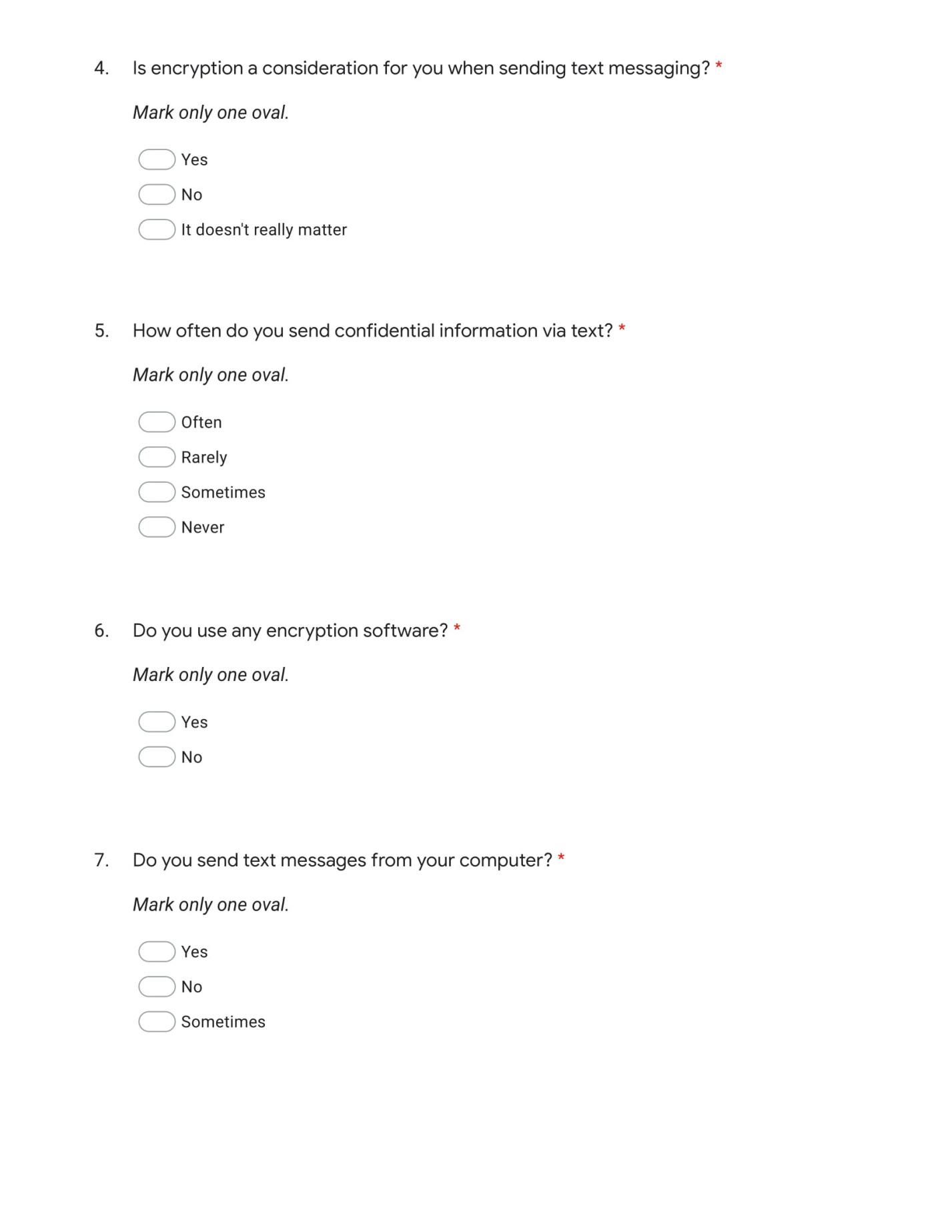
In future updates to the software, more features will be added such as:

* A networking component to allow for sending of encrypted files/text over the internet or Local Area Network(LAN)
* The ability to choose different encryption schemes
* The option to ecrypt files already on the computer

**APPENDIX**

Survey Questions





Research References

* Data collection Survey - https://docs.google.com/forms/d/e/1FAIpQLScBswUV0NLLMwvLpAm057D7I3BrY\_92zaIDbi6s8GxuVHSBJg/viewform?usp=sf\_link
* What is Encryption - <https://searchsecurity.techtarget.com/definition/encryption>
* Types of Encryption - <https://www.toptenreviews.com/software/articles/encryption-algorithms/>
* Public Key Cryptography –   
  <https://www.globalsign.com/en/ssl-information-center/what-is-public-key-cryptography/>
* RSA Algorithm -   
  <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

Programming References

* ASCII Character codes - <https://ascii.cl/>
* Convert char to ASCII in C - <https://stackoverflow.com/questions/1472581/printing-chars-and-their-ascii-code-in-c>
* Password Hiding - <https://www.engineersgarage.com/c-language-programs/user-authentication-access-secured-resources>
* C Color Codes - <https://www.programmingsimplified.com/c/graphics.h/colors>

Other References

* Creating auto-run files - <https://msdn.microsoft.com/en-us/library/windows/desktop/cc144206(v=vs.85).aspx>
* Install Creator - <https://install-creator.en.uptodown.com/windows>