

Password Encryption

Fundamentals of C Programming



May 15, 2019

Johnson Nguyen 12620136, Aamir Faiaz 12623195, Howin Tam 13556497, Divya Cyril 13192000, Eleftherios Asiminaris 12889097

Contents

[Introduction 2](#_Toc10235619)

[Objective 2](#_Toc10235620)

[Scope 2](#_Toc10235621)

[Program Features and Design 2](#_Toc10235622)

[Screen Layout Design 2](#_Toc10235623)

[Main Menu 2](#_Toc10235624)

[Creation of AMS account 3](#_Toc10235625)

[Login Menu 3](#_Toc10235626)

[Login 4](#_Toc10235627)

[Forgot Password 4](#_Toc10235628)

[User Menu 4](#_Toc10235629)

[Change Password 4](#_Toc10235630)

[Searching 5](#_Toc10235631)

[Saving and Loading 5](#_Toc10235632)

[Password Encryption and Decryption 5](#_Toc10235633)

[Account Deletion 6](#_Toc10235634)

[Debug Mode 6](#_Toc10235635)

[Test Function 6](#_Toc10235636)

[Operation Details 6](#_Toc10235637)

[Critical Thinking 7](#_Toc10235638)

[Integration of multiple files 7](#_Toc10235639)

[Scoping issues 7](#_Toc10235640)

[Input issues 8](#_Toc10235641)

[Encryption Issues 8](#_Toc10235642)

[Crashing Issues 8](#_Toc10235643)

[Saving /Loading issues 8](#_Toc10235644)

[Conclusion and Recommendation 8](#_Toc10235645)

[REFERENCES 8](#_Toc10235646)

[APPENDICES 9](#_Toc10235647)

[APPENDIX 1 9](#_Toc10235648)

[APPENDIX 2 10](#_Toc10235649)

[APPENDIX 3 11](#_Toc10235650)

[APPENDIX 4 11](#_Toc10235651)

[APPENDIX 5 12](#_Toc10235652)

# Introduction

This report outlines the design and implementation of a program that aims to provide a functioning and secure account management system. The report will overview the development, compare different encryption methods to process and store accounts data as well as discussions on handling bugs found during the development phase.

# Objective

The purpose of this project was to create a professional grade program that manipulates real-world data, using compression, encryption or translation. This has led to the creation of our program, the Account Management System (AMS). The purpose of the program is to provide a company with a readymade account management system, that handles the account creation and login. The passwords of these accounts are securely encrypted for security purposes. This offers the interested companies the opportunity to add their own functionality as the program has been left open ended for their custom use.

# Scope

The program we have created was designed to target companies looking for an open-ended program that will encrypt and decrypt information in the form of an account managing system. With the corporate world as the ideal market, the program maintains their private operations secure by further developing on a skeletal program we have provided. This includes account creation that encrypts the password, and a login feature.

The team was formed on 1st of May 2019 and was tasked to create a solution to a problem that we deal in the real virtual world. The primary deliverable of this project includes the C source file, a document on the design, implementation and testing of the program and finally an oral presentation in which an interactive forum will be conducted to develop a broader understanding of the programming procedure and overall design of the product. The entire project is scheduled to be completed by 31st of May.

# Program Features and Design

## Screen Layout Design

Due to the program being run on a CLI (Command Line Interface) terminal. The layout design is simple and easy to navigate. We have a 60-width screen and the text is restricted to this parameter, this is the standard and all following pages follows this standard. The design also includes a title on each major page which is separated by a dashed line this provides an aesthetically pleasing design for the user.

## Main Menu

The main menu is the first screen the users will see when they launch the program. The menu sets the standard for the user interface and so all other menus and functions should match its style and design. The main menu has the dash line at the top, and thus all following sections will print a similar line to indicate and replicate the idea of a new page. The main menu also has the “Main Menu” title printed once every time this page is visited. Since this function sets the standard, each corresponding page will have their own titles. The main menu consists of three options for the user: Create account/ log in to an existing account/exit the program, as seen in **figure 1**.

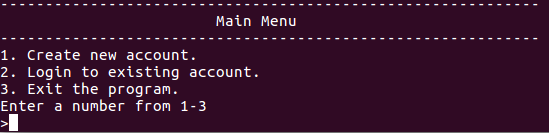


Figure Main Menu

## Creation of AMS account

The create account function is used when the user wishes to create an account which will take in a username, password and security answer. Similarly, with the main menu the section will contain a title “Create Account” which is again separated by the dash lines to indicate a new page. The user will be prompted to enter a username twice, that is limited to 10 characters and to avoid spaces. Next, the unique password will be entered in twice, whilst restricting the user to 10 characters. The set security question we have used is to ask, “What country were you born in?” this question will gather a unique answer from each user and will be used later if the user forgets their password. The question can easily be changed due to our programs modularity and at the wish of our customer, but at this point we have elected to provide an array of 56 characters for the length of the longest country name. Once the security answer has been entered, a confirmation page is produced so that the user can check information and thus elect to confirm (input = 1) or decline (input = 2) which will ultimately bring the user back to the username section to start again. The function also has a “back” feature built in, by typing “back” exactly, the user will be ejected from this page back to the previous which will be the main menu. The user must type back twice for the username to return to the main menu or if the user is at the stage of entering a password they will only need to type “back” once, this same scenario applies to the security question. An example of this can be seen in **figures 2** and **3.**

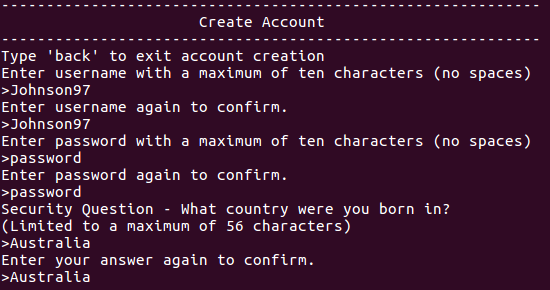


Figure Create Account

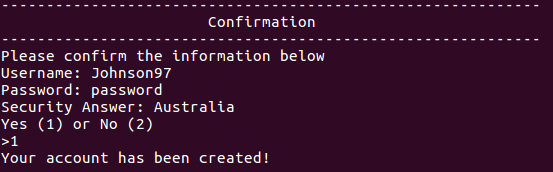


Figure Confirmation

## Login Menu

Once the user has a registered account, the login menu provides the following options: Login/Forget Password/Return to Main Menu. If the user wishes to login to an account the input is 1 where they will be taken to the login page. If the user has forgotten their password, select 2 so that they are taken to the forget password page. Select 3 if they wish to return to the main menu. The section also includes a login menu title separated by lines to give a pleasing look as well as to separate the information to provide a friendly expeirence.

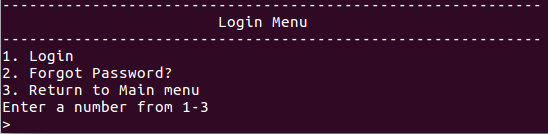


Figure Login Menu

### Login

The login is the first option in the login menu. In this page the user is required to enter in an existing username, followed by a password. If successful they will be sent to the next page where they will be greeted by their username. If unsuccessful they will be prompt to try again, the user is also given the option to go back to the login menu by typing “back”.

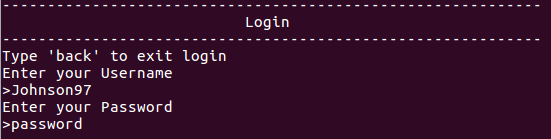


Figure Login

### Forgot Password

When in this page, the user is prompt to enter their username, followed by the answer to the security question, as seen in **figure 6**. Since each account has an associated answer this will be used to identify if the correct user is operating. Once the information is successfully entered, the user will be sent to the Change Password screen as seen in **figure 8**. Whilst in the forgot password screen, at any time the user can type “back” to go back to the previous screen.

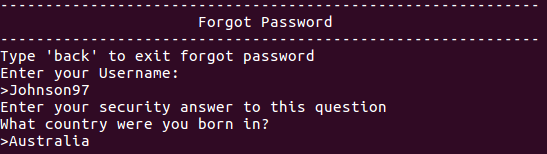


Figure Forgot Password

## User Menu

Once the user has successfully logged in, they will be greeted by their username. They are also given three options: Change Password/Delete Account/Logout as seen in **figure 7**. If the user wishes to change their password, they would enter 1. If they want to delete their account, they would enter 2 and to logout they would enter 3. Since this program provides the steppingstone for a company, they can easily implement their own features in this section.

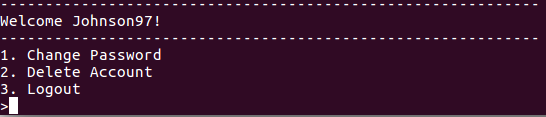


Figure User Menu

### Change Password

If the user is in this screen, they have either entered it by the User Menu or sent from the forgot password. In this menu user is asked to simply enter their new password twice and then asked to confirm the new password. The user can input 1 to accept or 2 to decline which will start the process all over again. At any point the user can type “back” to return to the previous page.

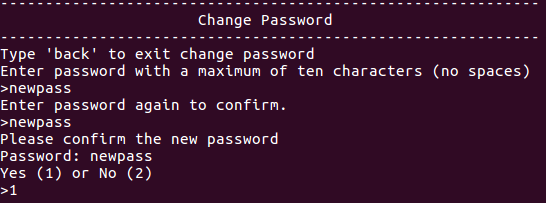


Figure Change Password

## Searching

Our program has a searching feature included into the program. When accounts are stored, they are stored in an array of structure with member arrays that represent a structure type, these include username, password, security answer and a key. Please refer to **figure** in the appendix. This means that each account will have an associated username, password, security answer and key. The searching function is implemented when the user attempts to login, the program will identify the username entered and compare the user entered password to its associated password. If the two match the login will be successful. In the forgot password, the program will identify the username entered and compare the user entered security answer to the associated answer, if the two match the program will perform its corresponding output.

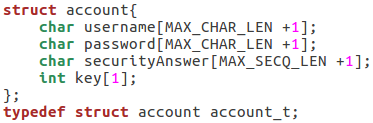


Figure Array of Structure

## Saving and Loading

To ensure user accounts and details are up to date, and can be accessed later, our program constantly saves any changes into the database file. Further, when the program initiates, it will first check whether there is a database file called AMS\_Accounts present. If this is true, it will begin loading accounts into the amsAccounts array, where it is ready to be accessed and changed. By saving it in a specific orientation, our program can properly decipher what it is reading, and where it should be saved in memory too. It will also count how many accounts are saved into the array, so that the creation of new accounts will not overwrite saved items. On the other hand, if no file is found, it will create an empty file, so that accounts can be saved.

## Password Encryption and Decryption

An important feature in our program is the encryption of the password, we have elected to create a Caesar cipher approach. This means each character in the password is shifted by the value of our key. During the creation of an account the program will take the length of characters in the username and use that as the key, in addition we have created a random number generator that chooses a number between 0 and 20 to be added to the key. This approach ensures that the key is as unique as possible so that we can securely encrypt our passwords. We also have a decryption feature that will decrypt the password so that it can compare the stored password with the user entered password when attempting to login. To do this we simply take the stored key and minus the password characters with the key value.

## Account Deletion

Our program has the option to delete accounts that are saved in the accounts array. It will ask for user input, then use exact search to locate which element array the account is saved in. When the user confirms account deletion, the program will replace all parts of the account, such as username, password, key and security answer with NULL values. This is used to ensure when it is used for new account creation, there are no left-over characters. The program will also shuffle the accounts to fill up the empty space that the account deletion created, therefore allowing new accounts to be added.

E.g, If Account in position 3 is deleted, the Account in position 4 and position 5 will move into position 3 and position 4 respectively.

## Debug Mode

Our program included a debug mode. This mode allowed us to compile the code whilst printing useful information that helped us understand how our program was behaving as we developed the code. Using the “#define DEBUG” at the beginning of our code allowed us to use “#ifdef DEBUG” and #endif, the way this worked was we would put the lines of code such as printing values of our variables between these two lines of code as seen in **figure 10** when the “#define DEBUG is commented out the lines of code will not compile thus producing a debug mode and a normal mode. The printing of these variables was extremely useful in debugging issues and understanding if we were on the right track, as we usually ran into scope issues as discussed in our issues section of the report.



Figure use of debug mode

### Test Function

Part of the debug mode was our testFunction, this function in our code was used to print the account usernames, password and answer so that we could see if our save and loading was working. This function was extremely helpful after we had created multiple accounts and needed to display the corresponding information. The function also allowed us check if our decryption of password was working. This function when debug mode is on can be accessed in the main menu by inputting 4, as this was the main page with the largest activity when testing our code allowing for ease of access.

# Operation Details

When the program is initialized the database file and first main menu is loaded. Then the user has 2 options to choose from: Create Account and Login to Account. Depending on which option is picked that process will be loaded and the parameters set. The user input will be saved to the database file or the user input will be cross checked with the database file. Please refer to **appendix 1 – 3**.

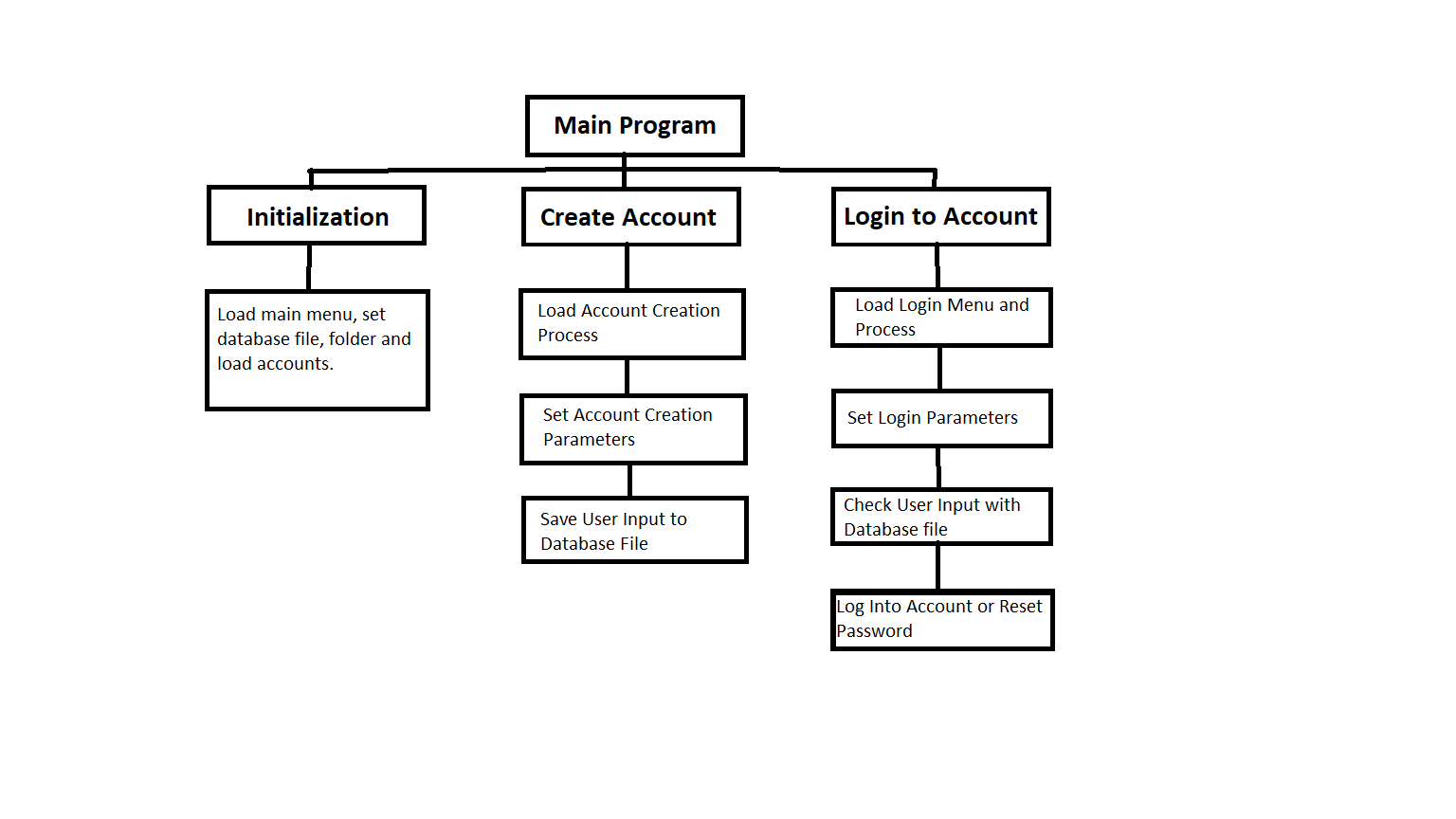


Figure Functional Hierarchy Diagram of Main Program

# Critical Thinking

There were many issues that we encountered when developing the code for the program. A majority of the time nothing worked as intended when initially implemented and required constant review and debug of the code. Below are a few of many issues that were present during this project.

## Integration of multiple files

When initially starting the project we had discussed ways in which we could best work on the program of the project. It was agreed that each member was to write a part of each function which meant we would have multiple C files, when the time came to integrate each function together each person had different variable names which made the task tedious to combine them into a single file. We discussed the use multiple IDE’s, github was also discussed however ultimately not all members had access to a terminal based system and since the assignment required it to be run on this format we decided to share the C files on a google drive, it was easy as everyone was familiar with it. Ultimately the easiest way to ensure our code followed a similar style was to have multiple versions of code and each time a feature or function was added, a new version is made with a different name, as seen in **figure** in the appendix. This ensured that code that was added and untested would not affect the already working C file. After a substantial amount of code was developed two members would continue to finish it off as well as debug whilst other members worked on the report.

## Scoping issues

When implementing functions within functions, we had issues in which values that were returned by the inner function did not reach beyond its scope, causing unexpected values to be used in our logic. There were two methods that we decided that could rectify this issue. Use pointers to send the value directly into the memory location. Or to write a majority of our features into one function, this would result in less functions and ultimately create less scope issues for us to handle. Our chosen method was to return the value multiple times. This was decided upon, as the value that is returned from the inner function was also used in the outer function. Ultimately, it allowed the returned value to be used within all layers of code, defeating our scoping issue.

## Input issues

Our Security Question Answer could not accept spaces, and instead only saved everything before the space, and nothing after. The chosen method that we decided to implement was the use of arguments within the scanf function. “%57[^n]” is an argument used in this function, which means it will take up to 57 characters until it reaches a “/n”. This means it will also save and read white spaces in the answer.

## Encryption Issues

Instead of encrypting our passwords properly, random characters would end up in the password array e.g we encountered an issue with lowercase a in which it would print lowercase b.3 instead of lowercase b by itself. Our chosen method for rectifying this issue was the testing of arguments. It was discovered that the encryption function was also changing the values of integers that were located next to the password array in memory. As such, by ensuring only the proper element of the array was encrypted and nothing else, it eliminated the random chance of incorrect values also saving into the password array.

## Crashing Issues

Our program had another issue when it was trying to save into the database file and the password array. When a password length of higher than 10 was inputted, it result into a stacking crash error. Further, random letters were saving into the database file, which would cause the overall orientation and organisation of the variables to change, affecting how they were saved in memory when it was loaded. In order to resolve this, we needed to conduct a large variety of testing. This led to our discovery of a piece of code trying to save 11 array elements into an array with size 10. By increasing the size of the array, this quickly solved the stacking issue and the saving issue.

## Saving /Loading issues

Finally we had an issue in which when the program loaded the accounts from database, it would read an additional account, meaning if there was 3 accounts, it would say there was four, but the last one would be empty. In order to fix this issue we moved the location in which the EOF if statement would stop the loading of accounts. By stopping the while function before it reached the end of the loading, it meant that it would not save the incomplete data into the account array.jjjj

# Conclusion and Recommendation

In summary, the aim of this project is to create an Account Management System (AMS). This is specifically designed to create, store and delete user login details for a company. Through the incorporation of encryption, the program created provide a digital safe house for all the employees personal details. Even though there were various issues in the code at the beginning, towards the end the of the project all the issues were resolved. Therefore, resulting in a fully functioning program that creates, saves and delete user login details.

# REFERENCES

Agarwal H, n.d, *XOR Cipher*, Geeksforgeeks, viewed 15th May 2019

<<https://www.geeksforgeeks.org/xor-cipher/>>

Lyons J, 2012, *Caesar Cipher*, Practica Cryptography, viewed 18th May 2019

<http://practicalcryptography.com/ciphers/caesar-cipher/>

cplusplus.com, n.d, *Tutorials,* viewed 30th May 2019

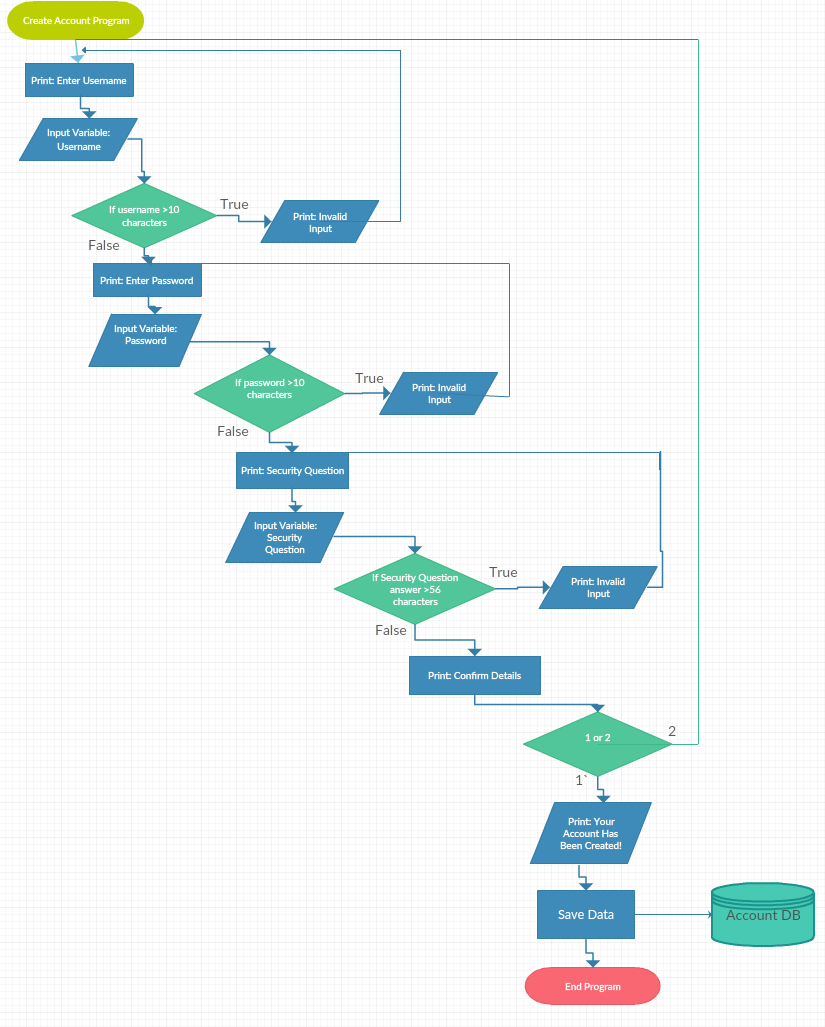
<<http://www.cplusplus.com/doc/>>

Learn Cryptography, 2019, *Caesar Cipher,* viewed 18th May 2019

<<https://learncryptography.com/classical-encryption/caesar-cipher>>

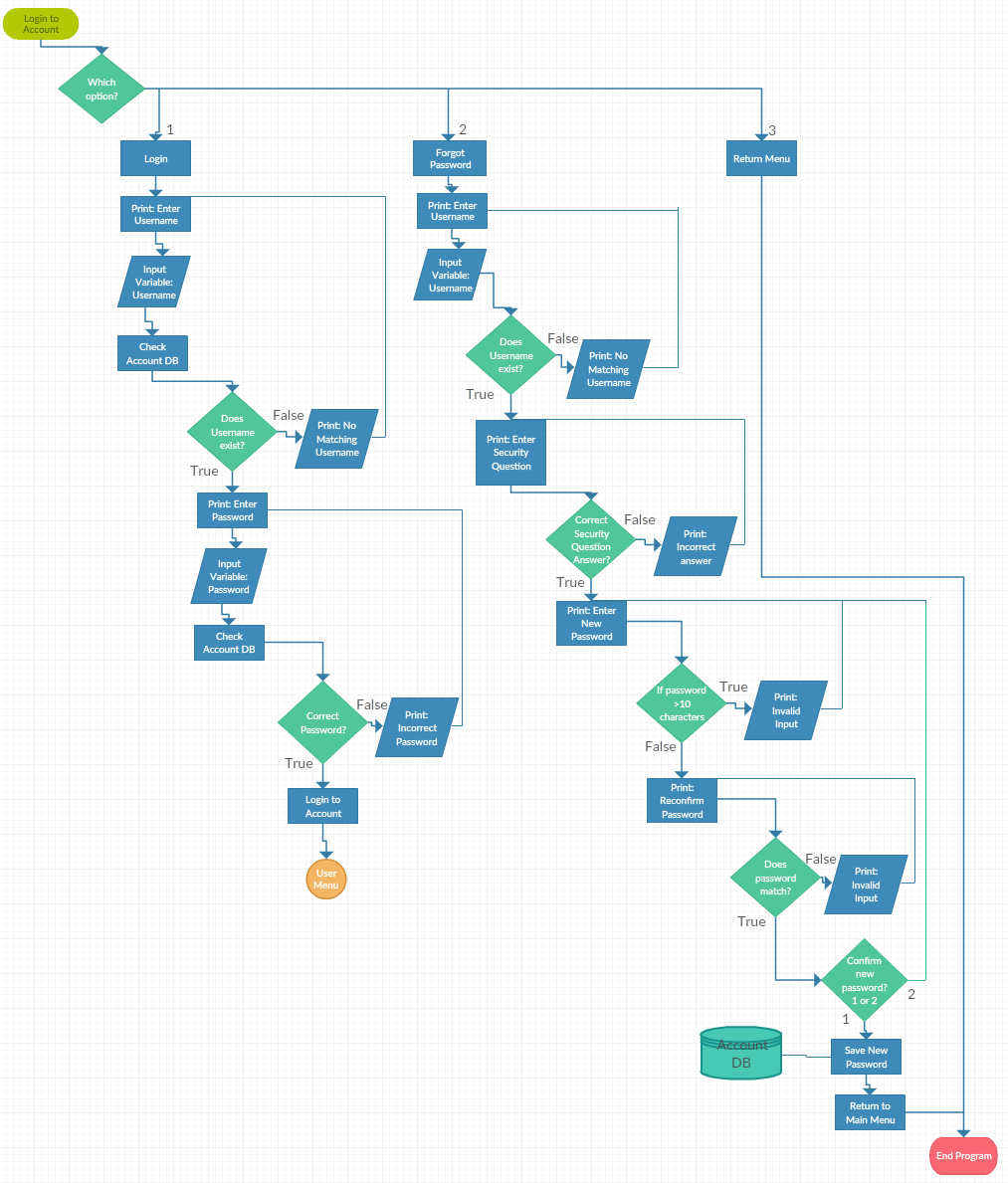
# APPENDICES

## APPENDIX 1



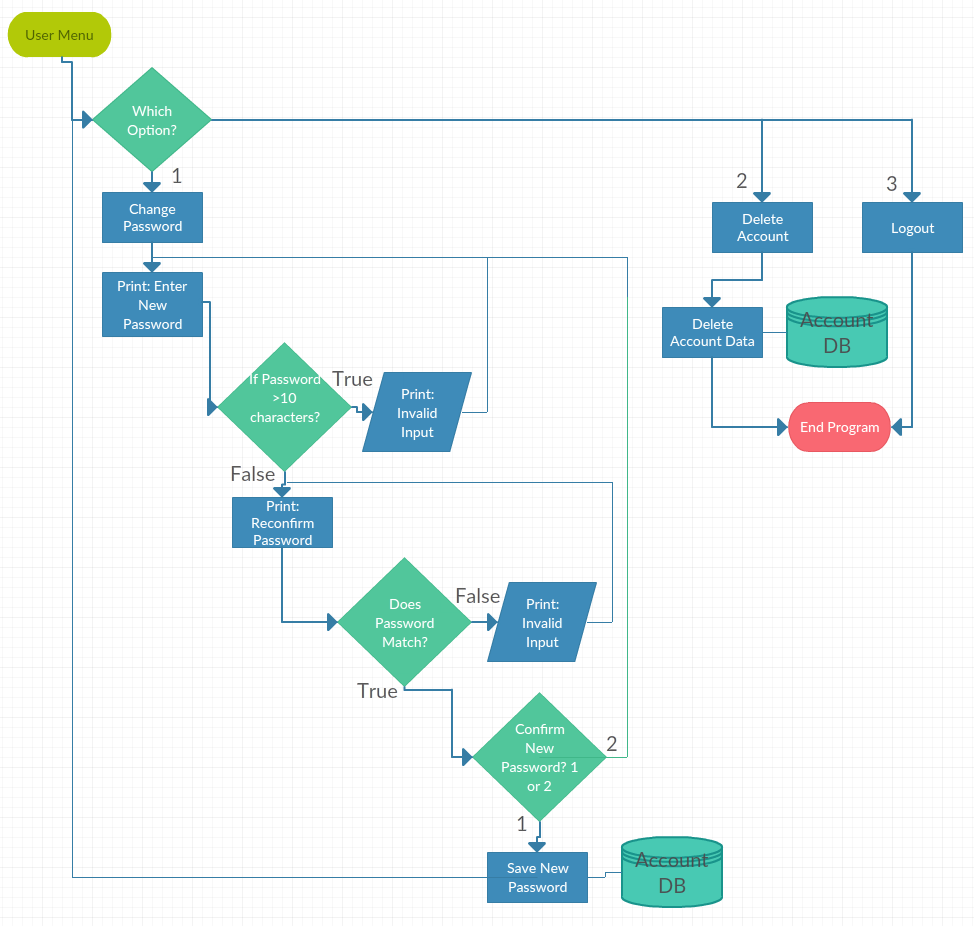
Appendix

## APPENDIX 2



Appendix

## APPENDIX 3



Appendix

## APPENDIX 4

Author: A

Contributor: C

Table Contribution Table Report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Report Sections | Johnson | Howin | Aamir | Divya | Eleftherios |
| Objective | C | A | C | C | C |
| Scope | C | A | C | A | C |
| Program Features | A | C | C | A | C |
| Flowcharts | C |  |  |  | A |
| Operations Details | C |  | C |  | A |
| Critical Thinking | C | C | C | A | C |

Table Contribution Table Code

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Code Functions | Johnson | Howin | Aamir | Divya | Eleftherios |
| initialiseProg | C | A |  |  |  |
| testFunction | A | C |  |  |  |
| print\_menu | C | C |  | A |  |
| create\_account | C | A |  | C |  |
| obtainCred | C | A | C | C |  |
| obtainKey | A | C | C |  |  |
| obtainSecurityQ | A | C |  | C |  |
| confirmDetails | A | C | C |  |  |
| amsStore | C | A |  |  |  |
| checkDup | C | A |  |  |  |
| verifyCredentials | C | A | C |  |  |
| saveProg | C | A |  |  |  |
| loginMenu | C | C | A | C |  |
| enterUser | A | C |  |  |  |
| encryption | A | C | C |  |  |
| Decryption | A | C | C |  |  |
| enterPass | A | C | C |  |  |
| forgetPassword | A | C |  |  |  |
| deleteAccount | C | A |  |  |  |
| shuffleArray | C | A |  |  |  |
| changePassword | A | C | C |  |  |
| successLogin | A | C |  |  |  |
| DEBUGGING | Y | Y | Y | Y |  |
| backFunc |  | A |  |  |  |

## APPENDIX 5

