

Programming Project

Pritam Gurung

Content

Table of Contents

1. Introduction	3
2. Program Design	3
2.1 Screen Layout	3
2.2 Data Storage Design	4
2.3 Skeleton Code	5
Nurse Program	5
For Consultant Programming	10
3. Implementation	16
3.1 Nurse Program	16
3.2 Consultant Program	17
3.3 Password File Maintenance	17
3.4 Data Encryption Techniques	18
4. Test Plans	18
4.1 Set Up Of Nurse Program	18
4.2 Set Up Of Consultant Program	19
4.3 What Works and Doesn't Work Nurse Program	20
4.4 What works and Doesn't work for Consultant Program	21
5. Evaluation (451 words)	21
6. My coding	23
6.1 Nurse	23
6.2 Consulting	28

1. Introduction

In this document, it presents two software programs designed to assist both nurses and consultants with their responsibilities. These programs are designed for the efficiency and security of patient data management for the nurse and consultant. The first program is designed for the nurses, allowing them to enter and store patient information. The second program is designed for a consultant, which allows them to secure access to these patient records for review and to add additional comments. Both programs are designed to secure data handling and ensure privacy security and integrity in managing confidential patients' information.

2. Program Design

2.1 Screen Layout

The Nurse Program operate through the command-line interface and begins with a clear, user-concise prompt which requests the following five nurse verified by the system for their ID and password. Once entered successfully, a series of clear and concise prompts asks the nurse to input the patient's details. This prompt asks for the patient's following, first name, last name, date of birth, height, weight, waist measurement and comments to add about the patient's condition with each step being accurate, concise, and self-intuitive for the nurse to navigate through the system as shown in Figure 1 showing the application of the system. The user interface is straightforward, ensuring that the nurses can enter the patient's details correctly, minimising the risks of inputting wrong data into the system. After inputting the patient's details, a confirmation message is shown confirming to the nurses that the patient's details have been stored/processed successfully can also be shown in Figure 1.

```
Nurse ID: BrianA
Password: abcdefg
Enter the first name of patient: Joshua
Enter the last name of patient: Ward
Enter the patient's DOB (DDMMYYYY): 27122003
Enter the height (cm): 180
Enter the weight (kg): 90
Enter the waist measurement (cm): 90
Enter comment: Stomach issues
Patient data has been processed successfully.
```

Figure 1 (nurse program) shows the program asking for nurse credentials, patient details and data being processed/stored.

The Consultant Program is also managed through a command-line interface, asking the consultant for its ID and password. After entering the consultant's details, the consultant is prompted to enter the patient's last name and date of birth, which is used to retrieve and display the patient's last record. The program then allows the consultant to add comments to the patient's file. This program is intuitive and easy to use, backed up by its simplicity and colour combination of black and white, providing efficient navigation for the consultant. Figure 2 shows the demonstration of these applications.

```

Consultant ID: StellaPryor
Password: password
Consultant authenticated successfully.
Enter patient's last name: Ward
Enter patient's DOB (DDMMYYYY): 27122003
Patient Name: Josh[a =ard
DOB: 27122003
Enter new comments: Stomach issue has been treated
Patient record updated successfully.

```

Figure 2 (consultant program) shows the program asking for consultant credentials and asking the consultant for a new comment, after inputting the patient's details.

2.2 Data Storage Design

The data storage design for both the nurse and consultant programs uses arrays and file storage to secure and maintain nurse and consultant information. In the nurse program shown in Image A, a 'struct' called 'Patient' is used to store patient data, including personal details like names and date of birth, measurement of height, weight and waist and comments that can be included on patients health. This information is then stored in a file named after patient's last name and date of birth, ensuring it's easier to find for each record and file extension '.aow' is used to represent Action on Weight Charity.

For consultant program, a similar approach is taken, with the addition of a 'Consultant' 'struct' to manage consultant login credentials. Passwords for both nurses and consultants are encrypted using Caesar cipher before storage, increasing security. Cipher employs a shift of seven character to obfuscate the data with 'char caesarCipher(char c, int shift)' and 'void stringEncrypt(char *str, int shift)'

Functions handling the encryption process. The passwords are stored in encrypted format within an array of 'Nurse' and 'Consultant' structure. These are shown in Image B

```

typedef struct Patient
{
    char firstName[NameSize];
    char lastName[NameSize];
    char dateOfBirth[DobFormatSize];
    float heightInCm;
    float weightInKg;
    float waistCm;
    char comment[CommentSpace];
} Patient;

typedef struct Nurse
{
    char id[IdLength];
    char passwordEncrypted[PasswordCapacity + 1];
} Nurse;

```

Image A shows struct of Patient and Nurse.

```

typedef struct Consultant
{
    char id[IdLength];
    char passwordEncrypted[PasswordCapacity + 1];
} Consultant;

```

```

void stringEncrypt(char *str, int shift)
{
    while (*str)
    {
        *str = caesarCipher(*str, shift);
        str++;
    }
}

void stringDecrypt(char *str, int shift)
{
    while (*str)
    {
        *str = caesarCipher(*str, -shift);
        str++;
    }
}

```

Image B Shows consultant struct and function.

2.3 Skeleton Code

Nurse Program

```
/* *****  
* nurse_program.c  
* Programming a nurse program to input patients details  
* Version a  
* Pritam Gurung  
* ***** */  
  
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
#include <stdlib.h>  
  
#define IdLength 50  
#define PasswordCapacity 8  
#define CipherShift 7  
#define NameSize 50  
#define NurseTotal 5 /* Define the total number of nurses */  
#define DobFormatSize 9  
#define CommentSpace 100  
  
typedef struct Patient  
{  
    char firstName[NameSize];  
    char lastName[NameSize];  
    char dateOfBirth[DobFormatSize];
```

```

float heightInCm;
float weightInKg;
float waistCm;
char comment[CommentSpace];
} Patient;

typedef struct Nurse
{
    char id[IdLength];
    char passwordEncrypted[PasswordCapacity + 1];
} Nurse;

/* Function prototypes */
char caesarCipher(char c, int shift);
void stringEncrypt(char *str, int shift);
void passwordEncrypt(char *password);
int loginNurse(Nurse nurseArray[], const char *id, const char
*password);
void detailsPatientGet(Patient *patient);
void detailsPatientEncrypt(Patient *patient);
void filePatientWrite(const Patient *patient);
void getInputNurseCredentials(char *id, char *password);
void processPatientData(Nurse nurseArray[], const char *id, const
char *password);

/* Caesar cipher function for encryption */
char caesarCipher(char c, int shift)
{
    /* Encryption logic */
}

```

```
/* Function to encrypt a string using the caesar cipher */
void stringEncrypt(char *str, int shift)
{
    /* Encrypting the string */
}

/* Function to encrypt a password */
void passwordEncrypt(char *password)
{
    /* Encrypting the password */
}

/* Function to check nurse login credentials */
int loginNurse(Nurse nurseArray[], const char *id, const char
*password)
{
    /* Checking nurse credentials */
}

/* Function to input patient details */
void detailsPatientGet(Patient *patient)
{
    /* Getting patient details */
}

/* Function to encrypt patient details */
void detailsPatientEncrypt(Patient *patient)
{
    /* Encrypting patient details */
}
```

```

/* Function to write patient data to a file */
void filePatientWrite(const Patient *patient)
{
    /* Writing patient data to file */
}

/* Function to get Nurse ID and password */
void getInputNurseCredentials(char *id, char *password)
{
    printf("Nurse ID: ");
    scanf("%49s", id);
    printf("Password: ");
    scanf("%7s", password);
    passwordEncrypt(password);
}

/* Function to process patient data */
void processPatientData(Nurse nurseArray[], const char *id, const
char *password)
{
    if (loginNurse(nurseArray, id, password))
    {
        Patient newPatient;
        detailsPatientGet(&newPatient);
        detailsPatientEncrypt(&newPatient);
        filePatientWrite(&newPatient);
        printf("Patient data has been processed successfully.\n");
    } else
    {

```



```

        printf("Access has been denied due to incorrect ID or
password.\n");
        exit(EXIT_FAILURE);
    }
}

/* Main function with simplified task structure */
int main()
{
    /* Array of nurse structure */
    Nurse nurseArray[NurseTotal] =
    {
        {"BrianA", "hijklmn"},
        {"HelenA", "opqrstu"},
        {"IanC", "vwxyzab"},
        {"DeniseM", "cdefghi"},
        {"JakobH", "jklmnop"}
    };

    /* Variables for ID and password input */
    char id[IdLength];
    char password[PasswordCapacity];

    /* Getting nurse details and processing patient data */
    getInputNurseCredentials(id, password);
    processPatientData(nurseArray, id, password);

    return 0;
}

```

For Consultant Programming

```
/* *****  
* consultant_program.c  
* Programming a consultant program to input patients details  
* by retrieving patient files from nurses  
* Version a  
* Pritam Gurung  
* ***** */  
  
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
#include <stdlib.h>  
  
#define IdLength 50  
#define PasswordCapacity 8  
#define CipherShift 7  
#define NameSize 50  
#define NurseTotal 5  
#define DobFormatSize 9  
#define CommentSpace 100  
  
typedef struct Patient  
{  
    char firstName[NameSize];  
    char lastName[NameSize];  
    char dateOfBirth[DobFormatSize];  
    float heightInCm;
```

```

    float weightInKg;
    float waistCm;
    char comment[CommentSpace];
} Patient;

typedef struct Consultant
{
    char id[IdLength];
    char passwordEncrypted[PasswordCapacity + 1];
} Consultant;

char caesarCipher(char c, int shift);
void stringEncrypt(char *str, int shift);
void stringDecrypt(char *str, int shift);
void passwordEncrypt(char *password);
int loginConsultant(Consultant *consultant, const char *id, const
char *password);
Patient readPatientFile(const char *filename);
void addConsultantComments(Patient *patient);
void filePatientWrite(const Patient *patient);

char caesarCipher(char c, int shift)
{
    if (isalpha(c))
    {
        char base = islower(c) ? 'a' : 'A';
        return (c - base + shift) % 26 + base;
    }
    if (isdigit(c))
    {
        return (c - '0' + shift) % 10 + '0';
    }
}

```

```
    }  
    return c;  
}  
  
void stringEncrypt(char *str, int shift)  
{  
    while (*str)  
    {  
        *str = caesarCipher(*str, shift);  
        str++;  
    }  
}  
  
void stringDecrypt(char *str, int shift)  
{  
    while (*str)  
    {  
        *str = caesarCipher(*str, -shift);  
        str++;  
    }  
}  
  
void passwordEncrypt(char *password)  
{  
    stringEncrypt(password, CipherShift);  
}  
  
int loginConsultant(Consultant *consultant, const char *id, const  
char *password)  
{
```

```

    return strcmp(consultant->id, id) == 0 && strcmp(consultant-
>passwordEncrypted, password) == 0;
}

```

```

Patient readPatientFile(const char *filename)
{

```

```

    Patient patient;

```

```

    FILE *file = fopen(filename, "r");

```

```

    if (!file)
    {

```

```

        perror("File opening failed");

```

```

        return patient;
    }

```

```

    fscanf(file, "%49s\n%49s\n%8s\n%f\n%f\n%f\n%99[^\n]",

```

```

        patient.firstName, patient.lastName, patient.dateOfBirth,

```

```

        &patient.heightInCm, &patient.weightInKg,

```

```

    &patient.waistCm, patient.comment);

```

```

    fclose(file);

```

```

    stringDecrypt(patient.firstName, CipherShift);

```

```

    stringDecrypt(patient.lastName, CipherShift);

```

```

    stringDecrypt(patient.comment, CipherShift);

```

```

    return patient;
}

```

```

void addConsultantComments(Patient *patient)
{

```

```

    printf("Enter new comments: ");

```

```

    scanf(" %99[^\n]", patient->comment);

```

```

}

```

```

void filePatientWrite(const Patient *patient)
{
    char filename[100];
    sprintf(filename, "%s%s_record.aow", patient->lastName, patient-
>dateOfBirth);
    FILE *file = fopen(filename, "w");
    if (!file)
    {
        perror("File opening failed");
        return;
    }

    Patient modifiablePatient = *patient;

    stringEncrypt(modifiablePatient.firstName, CipherShift);
    stringEncrypt(modifiablePatient.lastName, CipherShift);
    stringEncrypt(modifiablePatient.comment, CipherShift);

    fprintf(file, "%s\n%s\n%s\n%.2f\n%.2f\n%.2f\n%s\n",
            modifiablePatient.firstName, modifiablePatient.lastName,
modifiablePatient.dateOfBirth,
            modifiablePatient.heightInCm,
modifiablePatient.weightInKg, modifiablePatient.waistCm,
modifiablePatient.comment);

    fclose(file);
}

int main()
{

```

```

Consultant stellaPryor = {"StellaPryor", "whzzdvyk"};

char id[IdLength];
char password[PasswordCapacity + 1];
printf("Consultant ID: ");
scanf("%49s", id);
printf("Password: ");
scanf("%8s", password);
passwordEncrypt(password);

if (loginConsultant(&stellaPryor, id, password))
{
    printf("Consultant authenticated successfully.\n");
    char lastName[NameSize];
    char dob[DobFormatSize];
    printf("Enter patient's last name: ");
    scanf("%49s", lastName);
    printf("Enter patient's DOB (DDMMYYYY): ");
    scanf("%8s", dob);

    char filename[100];
    sprintf(filename, "%s%s_record.aow", lastName, dob);

    Patient existingPatient = readPatientFile(filename);

    if (strlen(existingPatient.firstName) == 0)
    {
        printf("Patient record not found.\n");
        exit(EXIT_FAILURE);
    }
}

```

```
    printf("Patient Name: %s %s\n", existingPatient.firstName,
existingPatient.lastName);
    printf("DOB: %s\n", existingPatient.dateOfBirth);
    addConsultantComments(&existingPatient);
    filePatientWrite(&existingPatient);

    printf("Patient record updated successfully.\n");
}
else
{
    printf("Access denied. Incorrect ID or password.\n");
    exit(EXIT_FAILURE);
}

return 0;
}
```

3. Implementation

3.1 Nurse Program

The nurse program is designed and programmed to collect and store patient's information securely and the nurses can input patient data, such as names, dates of birth, and medical measurement as shown in Image C. Furthermore, once details have been entered of patients, the information provided is encrypted and saved to patient files, which are named by 'lastnamedob_record.aow' to easily identify for consultants. The program also includes a secure login system, ensuring that only authorised 5 nursing staffs can access and enter patients data. This functionality is key to keeping patients details safe and to ensure to maintain integrity of data security.

```
Enter the first name of patient: Joshua
Enter the last name of patient: Ward
Enter the patient's DOB (DDMMYYYY): 27122003
Enter the height (cm): 180
Enter the weight (kg): 90
Enter the waist measurement (cm): 90
```

Image C, shows patients details being inputted into the system.

3.2 Consultant Program

The consultant program allows secure access to patient records for authorised medical consultants. Upon entering the credentials and getting access, the consultant can retrieve a specific patient's file using their last name and date of birth as identifiers like this 'lastnamedob_record.aow'. This program then decrypts and displays the patients's information, which allows the consultant to review and add comments to the patient's report. Any changes made are re-encrypted and saved back to the file, ensuring that the patient's data remains secure and confidential throughout the process. This can be shown in Image D.

```
Consultant ID: StellaPryor
Password: password
Consultant authenticated successfully.
Enter patient's last name: Ward
Enter patient's DOB (DDMMYYYY): 27122003
Patient Name: Josh[a =ard
DOB: 27122003
Enter new comments: Stomach issue has been treated
```

Image D shows the capabilities accessed by Consultants such as entering new comment and inputting patient's information to retrieve specific file.

3.3 Password File Maintenance

In both the nurse and consultant program, passwords are stored in an encrypted format so that when a new password is entered for example, it is encrypted using Caesar cipher method, which shifts each character by a determined number which makes it secure. These encrypted passwords are then saved in an array within the program. For maintenance and

authentication purposes, the programs compare the entered password against the encrypted versions to grant access. What this does is that it ensures that passwords are not stored in plain text which maintains security as well as data integrity.

3.4 Data Encryption Techniques

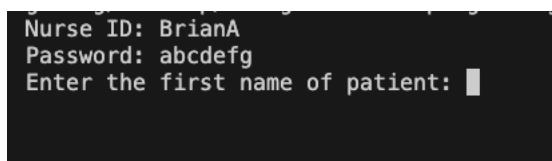
The program uses Caesar cipher for encryptions and decryptions as this method shifts each letter in the password by a set number of places down the alphabet. For decryption, it reverses the process which shifts the letter back to its original position and this simple method secures the password by transforming it into a coded format that is not easy to read, which is designed for high security.

4. Test Plans

In this section, I will highlight what works in both the nurse and consultant programs with detailed explanations as well as what doesn't work with images to aid visual learners.

4.1 Set Up Of Nurse Program

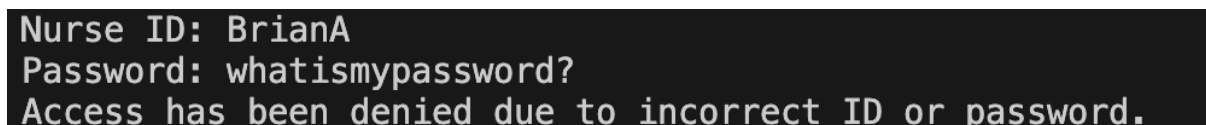
Upon entering the program, the program prompts the nurse to enter it's ID. The known ID for the program are 'BrianA', 'HelenA', 'IanC', 'DeniseM' and 'JakobH' which are the following ID that are needed. Followed by a password, each unique to different nurses for security purposes and for this demonstration, I'll show an screenshot of Brian entering his credentials into the program which should grant him access as shown in Figure a.



```
Nurse ID: BrianA
Password: abcdefg
Enter the first name of patient: █
```

Figure a shows Brian granted access.

This is a screenshot of Brian being denied access shown in Figure g. This shows the security capabilities of the programs as shown in Figure g,



```
Nurse ID: BrianA
Password: whatismypassword?
Access has been denied due to incorrect ID or password.
```

Figure g shows Brian being denied access upon entering its credentials.

Furthermore, upon entering the correct credentials, you are asked to enter the medical information of the patients, for demonstration let's call the patient Bob Mac, which is shown in Figure b.

```

Enter the first name of patient: Bob
Enter the last name of patient: Mac
Enter the patient's DOB (DDMMYYYY): 18091998
Enter the height (cm): 170
Enter the weight (kg): 100
Enter the waist measurement (cm): 80
Enter comment: Breathing Problem
Patient data has been processed successfully.

```

Figure b shows nurse entering the details of the patient 'Bob Mac' into the system where it's saved/processed for Consultant view.

To conclude, the set-up is really intuitive and easy to navigate around due to it's simplicity.

4.2 Set Up Of Consultant Program

Upon entering the program, the program prompts the consultant Stella to enter it's credentials to be granted access as shown in Figure c.

```

Consultant ID: Stella
Password: 

```

Figure C prompting Stella to enter it's credentials.

When Stella enters the wrong details, they will be denied access as shown in figure d.

```

Consultant ID: Stella
Password: mooley
Access denied. Incorrect ID or password.

```

Figure d showing access denied.

When Stella enter the right details, they will be granted access as shown in Figure e with a message confirming the credentials are correct.

```

Consultant ID: StellaPryor
Password: password
Consultant authenticated successfully.

```

Figure e shows Stella being granted access.

After being granted access, Stella the consultant is required by the system to enter it's the details of the patient and previously in the nurse program, Brian has inputted the patient's data of Bob Mac. After having entered Bobs, important detail like it's last name and date of birth, Stella the Consultant can now add comments on Bob Mac which could be on his condition as shown in figure f.

```

Consultant ID: StellaPryor
Password: password
Consultant authenticated successfully.
Enter patient's last name: Mac
Enter patient's DOB (DDMMYYYY): 18091998
Patient Name: Bob Mac
DOB: 18091998
Enter new comments: Bob has recovered.
Patient record updated successfully.

```

Figure f showing inputting patient's data and record being updated successfully.

To conclude, the Consultant program serves its purpose and it's intuitive to navigate around and further, by its simplicity, it's easy to use and friendly.

4.3 What Works and Doesn't Work Nurse Program

In the nurse program, the functionality for entering patient data works as expected. The program successfully prompts for and accepts input for the patient's first name, last name, date of birth, height, weight, waist measurements and comments. Once the information of the patient has been entered, it is processed and confirmed with a message that indicates data has been saved/processed as shown in Image E.

```

Nurse ID: BrianA
Password: abcdefg
Enter the first name of patient: Joshua
Enter the last name of patient: Ward
Enter the patient's DOB (DDMMYYYY): 27122003
Enter the height (cm): 180
Enter the weight (kg): 90
Enter the waist measurement (cm): 90
Enter comment: Stomach issues
Patient data has been processed successfully.

```

Image E shows the working functionality of the Nurse Program.

However, the validation of user input is not fully evident in the provided test log. While the program does appear to accept valid data, it's not clear from this test alone whether a program can correctly handle invalid or out-of-range inputs, such as incorrect date formats or non-numerical values for measurements. An additional test is a must to confirm the robustness of input validation. As shown in Image F, putting in incorrect values does not stop the users which are the nurses from finishing the program and does not show any messages to correct the user's mistakes or prevent users from typing incorrect values.

```

Nurse ID: HelenA
Password: hijklmn
Enter the first name of patient: 27394
Enter the last name of patient: 12312
Enter the patient's DOB (DDMMYYYY): 128371289371283
Enter the height (cm): Enter the weight (kg): 12391283912839
Enter the waist measurement (cm): adasdadsd
Enter comment: Patient data has been processed successfully.

```

Image F displays the flaws of this program, capable of accepting any values into the system which can be a risk in the medical profession.

4.4 What works and Doesn't work for Consultant Program

The consultant program's authentication system is functioning correctly as it should be as evidenced by the log shown in Image G. It successfully denied access when an incorrect ID or password was entered, demonstrating effective security measures however when provided with valid credentials, the program is granted access, and the consultant Stella, can retrieve and update a patient's record. The addition of new comments to the patient file, followed by an update confirmation message, shows the program's ability for data retrieval and modifications which are working as intended shown in Image H. The test log indicates that the program correctly decrypts the patient's name for display although there appears to be an issue in the decryption output, which could show an issue with character encoding. To conclude, further testing is required to ensure accuracy of this program as well as decrypting function.

```

C:\Program Files\Consultant Program\assignment for prog
Consultant ID: StellaPryor
Password: jdgdsd
Access denied. Incorrect ID or password.

```

Image G shows access being denied upon entering invalid credentials.

```

Program: C:\Program Files\Consultant Program\assignment for prog
Consultant ID: StellaPryor
Password: password
Consultant authenticated successfully.
Enter patient's last name: Ward
Enter patient's DOB (DDMMYYYY): 27122003
Patient Name: Josh[a =ard
DOB: 27122003
Enter new comments: Stomach issue has been treated
Patient record updated successfully.

```

Image H shows the features of Consultant Program upon entering credentials successfully.

5. Evaluation (451 words)

The development of the nurse and consultant program was aimed at using the secure management of patient data with robust functionalities and some areas that required improvements.

The nurse program was efficient, allowing inputs of patient data through the interface, which encrypts and writes to a named file. The use of structured data 'Patient' struct ensures that all information is captured by the system. In practice, the program handles data input effectively as evidenced by the successful processing of messages. However, the test logs do not show how the system would behave with invalid inputs, such as incorrect date formats or non-numerical values for measurements. This lack of input validation can lead to potential data integrity issues or crashes, which is an area of improvement I can learn from.

The consultant program's functionality complements the nurse's data entry by allowing access to the patient records for review and updates. The login authentication worked as intended and denied access on incorrect credentials and granted access upon the correct credential being entered. Once logged in, the consultant can access patient records. Moreover, there is a issue in the decryption output which indicates its flaws in the process or issue with character encoding. This flaw can interrupt readability of data and needs fixing.

Both programs utilise a Caesar cipher for encrypting passwords, which works well for security purposes. However, the method, while simple is not robust against modern cyber-attacks, for real world applications, a much-advanced security is needed. In terms of usage, each program provides clear prompts and feedback which is good but it lacks the ability to correct users and inform them a correct value to be entered for access, which I don't have implemented in the program and this could potentially cause issues such as crash.

During the development of my program, research was made on secure password handling and data encryption method which was sourced from YouTube tutorials and a very good website called [geeksforgeeks](#) and [w3school](#), making me employ these codes far more effectively with good understanding as well as class notes and lecture powerpoints. These resoruces helped me make the program possible. In terms of structure, both program is well-commented and structured, which helps understand the mechanism behind codes.

In conclusion, the nurse and consultant program do serve their intended purposes with operating functions as expected. The areas I could improve on include input validation and decryption accuracy which are critical for systems to read. Further research into advanced encryption methods would improve my system's robustness, making it more efficient. I have learnt a lot from this program and will continue to do so in Codecademy and hope to create a far more advanced project in the future to improve my coding skills.

6. My coding

6.1 Nurse

```

/* *****
* nurse_program.c
* Programming a nurse program to input patients details
* Version a
* Pritam Gurung
* ***** */

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#define IdLength 50
#define PasswordCapacity 8
#define CipherShift 7
#define NameSize 50
#define NurseTotal 5 /*Define the total number of nurses*/
#define DobFormatSize 9
#define CommentSpace 100

typedef struct Patient
{
    char firstName[NameSize];
    char lastName[NameSize];
    char dateOfBirth[DobFormatSize];
    float heightInCm;

```

```

float weightInKg;
float waistCm;
char comment[CommentSpace];
} Patient;

typedef struct Nurse
{
    char id[IdLength];
    char passwordEncrypted[PasswordCapacity + 1];
} Nurse;

char caesarCipher(char c, int shift);
void stringEncrypt(char *str, int shift);
void passwordEncrypt(char *password);
int loginNurse(Nurse nurseArray[], const char *id, const char
*password);
void detailsPatientGet(Patient *patient);
void detailsPatientEncrypt(Patient *patient);
void filePatientWrite(const Patient *patient);

char caesarCipher(char c, int shift)
{
    if (isalpha(c))
    {
        char base = islower(c) ? 'a' : 'A';
        return (c - base + shift) % 26 + base;
    }
    if (isdigit(c))
    {
        return (c - '0' + shift) % 10 + '0';
    }
}

```



```

    return c;
}

void stringEncrypt(char *str, int shift)
{
    while (*str)
    {
        *str = caesarCipher(*str, CipherShift);
        str++;
    }
}

void passwordEncrypt(char *password)
{
    stringEncrypt(password, CipherShift);
}

int loginNurse(Nurse nurseArray[], const char *id, const char
*password)
{
    for (int i = 0; i < NurseTotal; i++)
    {
        if (strcmp(nurseArray[i].id, id) == 0 &&
strcmp(nurseArray[i].passwordEncrypted, password) == 0)
        {
            return 1;
        }
    }
    return 0;
}

```

```

void detailsPatientGet(Patient *patient)
{
    printf("Enter the first name of patient: ");
    scanf("%49s", patient->firstName);
    printf("Enter the last name of patient: ");
    scanf("%49s", patient->lastName);
    printf("Enter the patient's DOB (DDMMYYYY): ");
    scanf("%8s", patient->dateOfBirth);
    printf("Enter the height (cm): ");
    scanf("%f", &patient->heightInCm);
    printf("Enter the weight (kg): ");
    scanf("%f", &patient->weightInKg);
    printf("Enter the waist measurement (cm): ");
    scanf("%f", &patient->waistCm);
    printf("Enter comment: ");
    scanf(" %99[^\n]", patient->comment);
}

void detailsPatientEncrypt(Patient *patient)
{
    stringEncrypt(patient->firstName, CipherShift);
    stringEncrypt(patient->lastName, CipherShift);
    stringEncrypt(patient->comment, CipherShift);
}

void filePatientWrite(const Patient *patient)
{
    char filename[100];
    sprintf(filename, "%s%s_record.aow", patient->lastName, patient->dateOfBirth);
    FILE *file = fopen(filename, "w");
}

```

```

if (!file)
{
    perror("File opening failed");
    exit(EXIT_FAILURE);
}
fprintf(file, "%s\n%s\n%s\n%.2f\n%.2f\n%.2f\n%s\n",
        patient->firstName, patient->lastName, patient->dateOfBirth,
        patient->heightInCm, patient->weightInKg, patient->
>waistCm, patient->comment);
fclose(file);
}

int main()
{
    /* Array of nurse structure */
    Nurse nurseArray[NurseTotal] =
    {
        {"BrianA", "hijklmn"},
        {"HelenA", "opqrstu"},
        {"IanC", "vwxyzab"},
        {"DeniseM", "cdefghi"},
        {"JakobH", "jklmnop"}
    };

    /* Variable for ID and password input */
    char id[IdLength];
    char password[PasswordCapacity];

    /* Nurse ID and password input */
    printf("Nurse ID: ");
    scanf("%49s", id);

```

```

printf("Password: ");
scanf("%7s", password);

/* Ecrption of entered password*/
passwordEncrypt(password);

/* Attempt login with provided details */
if (loginNurse(nurseArray, id, password))
{
    /* When login succesfully, get patients details */
    Patient newPatient;
    detailsPatientGet(&newPatient);
    detailsPatientEncrypt(&newPatient); /* Ecnrypts patients details */
    filePatientWrite(&newPatient); /* Patient details to a file */
    printf("Patient data has been processed successfully.\n");
}
else
{
    printf("Access has been denied due to incorrect ID or
password.\n");
    exit(EXIT_FAILURE);
}

return 0;
}

```

6.2 Consulting

```

/* *****
* consultant_program.c
* Programming a consultant program to input patients details
* by retrieving patient files from nurses
* Version a
* Pritam Gurung
* ***** */

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#define IdLength 50
#define PasswordCapacity 8
#define CipherShift 7
#define NameSize 50
#define NurseTotal 5
#define DobFormatSize 9
#define CommentSpace 100

typedef struct Patient
{
    char firstName[NameSize];
    char lastName[NameSize];
    char dateOfBirth[DobFormatSize];
    float heightInCm;
    float weightInKg;
    float waistCm;
    char comment[CommentSpace];
}

```

```

} Patient;

typedef struct Consultant
{
    char id[IdLength];
    char passwordEncrypted[PasswordCapacity + 1];
} Consultant;

char caesarCipher(char c, int shift);
void stringEncrypt(char *str, int shift);
void stringDecrypt(char *str, int shift);
void passwordEncrypt(char *password);
int loginConsultant(Consultant *consultant, const char *id, const
char *password);
Patient readPatientFile(const char *filename);
void addConsultantComments(Patient *patient);
void filePatientWrite(const Patient *patient);

char caesarCipher(char c, int shift)
{
    if (isalpha(c))
    {
        char base = islower(c) ? 'a' : 'A';
        return (c - base + shift) % 26 + base;
    }
    if (isdigit(c))
    {
        return (c - '0' + shift) % 10 + '0';
    }
    return c;
}

```

```
void stringEncrypt(char *str, int shift)
{
    while (*str)
    {
        *str = caesarCipher(*str, shift);
        str++;
    }
}

void stringDecrypt(char *str, int shift)
{
    while (*str)
    {
        *str = caesarCipher(*str, -shift);
        str++;
    }
}

void passwordEncrypt(char *password)
{
    stringEncrypt(password, CipherShift);
}

int loginConsultant(Consultant *consultant, const char *id, const
char *password)
{
    return strcmp(consultant->id, id) == 0 && strcmp(consultant-
>passwordEncrypted, password) == 0;
}
```

```

Patient readPatientFile(const char *filename)
{
    Patient patient;
    FILE *file = fopen(filename, "r");
    if (!file)
    {
        perror("File opening failed");
        return patient;
    }
    fscanf(file, "%49s\n%49s\n%8s\n%f\n%f\n%f\n%99[^\n]",
           patient.firstName, patient.lastName, patient.dateOfBirth,
           &patient.heightInCm, &patient.weightInKg,
    &patient.waistCm, patient.comment);
    fclose(file);

    stringDecrypt(patient.firstName, CipherShift);
    stringDecrypt(patient.lastName, CipherShift);
    stringDecrypt(patient.comment, CipherShift);

    return patient;
}

void addConsultantComments(Patient *patient)
{
    printf("Enter new comments: ");
    scanf(" %99[^\n]", patient->comment);
}

void filePatientWrite(const Patient *patient)
{
    char filename[100];

```



```

    sprintf(filename, "%s%s_record.aow", patient->lastName, patient-
>dateOfBirth);
    FILE *file = fopen(filename, "w");
    if (!file)
    {
        perror("File opening failed");
        return;
    }

    Patient modifiablePatient = *patient;

    stringEncrypt(modifiablePatient.firstName, CipherShift);
    stringEncrypt(modifiablePatient.lastName, CipherShift);
    stringEncrypt(modifiablePatient.comment, CipherShift);

    fprintf(file, "%s\n%s\n%s\n%.2f\n%.2f\n%.2f\n%s\n",
            modifiablePatient.firstName, modifiablePatient.lastName,
modifiablePatient.dateOfBirth,
            modifiablePatient.heightInCm,
modifiablePatient.weightInKg, modifiablePatient.waistCm,
modifiablePatient.comment);

    fclose(file);
}

int main()
{
    Consultant stellaPryor = {"StellaPryor", "whzzdvyk"};

    char id[IdLength];
    char password[PasswordCapacity + 1];

```

```

printf("Consultant ID: ");
scanf("%49s", id);
printf("Password: ");
scanf("%8s", password);
passwordEncrypt(password);

if (loginConsultant(&stellaPryor, id, password))
{
    printf("Consultant authenticated successfully.\n");
    char lastName[NameSize];
    char dob[DobFormatSize];
    printf("Enter patient's last name: ");
    scanf("%49s", lastName);
    printf("Enter patient's DOB (DDMMYYYY): ");
    scanf("%8s", dob);

    char filename[100];
    sprintf(filename, "%s%s_record.aow", lastName, dob);

    Patient existingPatient = readPatientFile(filename);

    if (strlen(existingPatient.firstName) == 0)
    {
        printf("Patient record not found.\n");
        exit(EXIT_FAILURE);
    }

    printf("Patient Name: %s %s\n", existingPatient.firstName,
existingPatient.lastName);
    printf("DOB: %s\n", existingPatient.dateOfBirth);
    addConsultantComments(&existingPatient);
}

```

```
filePatientWrite(&existingPatient);

printf("Patient record updated successfully.\n");
}
else
{
    printf("Access denied. Incorrect ID or password.\n");
    exit(EXIT_FAILURE);
}

return 0;
}
```