University of Nottingham
UK | CHINA | MALAYSIA

COMP1028 Programming and Algorithm

Session: Autumn 2024

Group Coursework

Group Name	**C Lovers**					
Group Members						
Name 1 (Leader)	**Jeffrey Teoh Dass**			**ID 1**	**20607904**	
Name 2	**Ibuki Furusho**			**ID 2**	**20716304**	
Name 3	**Ong Wei Xuan**			**ID 3**	**20617520**	
Name 4	**Shirley Ng Shuet Ling**			**ID 4**	**20618865**	
Name 5				**ID 5**		
Marks **Total (100)**	Program functionality and code quality (50)	Usability (10)	Comments(5)	Documentation (5)	Presentation & Demo (Recording) (15)	Bonus Feature (15)
Leader Sign	*Jeffrey*					
Date	**2-12-2024**					

Project Title: Contact Manager

1.0 Introduction

The **Contact Manager** is a comprehensive program designed to manage and organize contact information effectively. It provides a user-friendly interface to perform various operations on contacts while ensuring data security and integrity.

Purpose:

The primary goal of the Contact Manager is to simplify the process of storing, retrieving, and managing personal or professional contacts. It ensures that users can securely save their data while maintaining quick access and accurate management.

Key Features:

1. **Data Validation:**
    - Ensures that user inputs are valid (e.g., proper name, phone number, email formats).
    - Handles edge cases to prevent invalid or malformed entries.
2. **Batch Operations:**
    - Supports multiple operations such as adding, deleting, editing, and displaying contacts in one session.
3. **Add, Delete, Edit, and Display Contacts:**
    - Provides functionality to add new contacts, remove existing ones, or modify their details.
    - Displays the full contact list or filtered results.
4. **Merge Sort for Sorting Contacts:**
    - Sorts contacts based on various criteria like name, phone number, email, or the most recently added entry.
    - Ensures efficient ordering for easy navigation.
5. **Binary Search for Quick Retrieval:**
    - Quickly finds contacts using binary search by partial matches in names, phone numbers, or email addresses.
    - Provides fast and efficient search capabilities even for large datasets.
6. **Encryption for Data Security:**

- Contacts are encrypted before being saved to the file to prevent unauthorized access.
- Uses a Caesar cipher to obfuscate phone numbers and emails.

7. **Decryption Upon Login:**
   - Decrypts the data file upon login to allow access to the original contact information.
   - Ensures secure storage without compromising usability.

This project combines robust security measures, efficient algorithms, and practical functionality to create a reliable tool for managing personal and professional contacts.

2.0 Installation

Step-by-step instructions on how to install and set up the project.

```
# Change directory

cd Downloads (any directory where the downloaded source code is)


# Compile the program

gcc -o ContactManager.c
```

3.0 Usage

Instructions on how to run the program, including any command-line arguments or options.

```
# Run the program

./ContactManager [options]
```

4.0 Features

The following are the main features of our program with the corresponding screenshots.

- Add (batch operation)

```
Enter the number of what you would like to do, (-1 to exit): 1
Enter the name: Ashley
Enter the phone number (number must be exactly 10 digits): 0123456789
Enter the email: ashley@gmail.com

Contact added successfully

Would you like to continue adding contacts?(n to stop / any other char to continue): y

Enter the name: Beatrice
Enter the phone number (number must be exactly 10 digits): 0198765432
Enter the email: beatrice@gmail.com

Contact added successfully
```

Figure 1 shows that the program will prompt user if they want to add multiple contacts.

- Delete (batch)

```
================================================
Contact 1
        Name: Ashley
        Phone number: 0123456789
        Email: ashley@gmail.com


Contact 2
        Name: Beatrice
        Phone number: 0198765432
        Email: beatrice@gmail.com


Contact 3
        Name: Casey
        Phone number: 0134567298
        Email: casey@gmail.com


Contact 4
        Name: Delancy
        Phone number: 0178923465
        Email: delancy@gmail.com

================================================
Enter the index for the contact you would like to delete: 1

Contact successfully deleted

Would you like to continue deleting contacts?(n to stop / any other char to continue): y
```

Figure 2.1 The program will prompt the user whether they want to continue deleting contacts

```
==================================================
Contact 1
        Name: Beatrice
        Phone number: 0198765432
        Email: beatrice@gmail.com


Contact 2
        Name: Casey
        Phone number: 0134567298
        Email: casey@gmail.com


Contact 3
        Name: Delancy
        Phone number: 0178923465
        Email: delancy@gmail.com

==================================================
Enter the index for the contact you would like to delete: 1

Contact successfully deleted

Would you like to continue deleting contacts?(n to stop / any other char to continue):
```

Figure 2.2 The program will then display the contact list with the updated contacts after deleting the first input, then prompt user if they want to delete. The program will stop prompting the user to continue deleting until user enters "n"

```
==================================================
Contact 1
        Name: Delancy
        Phone number: 0178923465
        Email: delancy@gmail.com

==================================================
Enter the index for the contact you would like to delete: 1

Contact successfully deleted

Would you like to continue deleting contacts?(n to stop / any other char to continue): y

There are no contacts to delete, returning to main menu

**********Contact Manager**********
                1. Add Contact
                2. Edit Contact
                3. Delete Contact
                4. Display Contacts
                5. Sort Contacts
                6. Search Contact
                7. Save Contacts
**************************************************
Enter the number of what you would like to do, (-1 to exit):
```

Figure 2.3 Once all the contacts have been deleted in the batch delete function, the program will display that there are no contacts in the contacts list and return to the main menu

- Edit (batch)

```
Enter the contact index to edit: 3

Contact 3
        Name: c
        Phone number: 0136987542
        Email: keshi@gmail.com

What would you like to edit?:
 [1] Name
 [2] Phone Number
 [3] Email

Enter your choice: 1
Enter the new name: Keshi

Name successfully edited!

Would you like to continue editing contacts? (n to stop / any other char to continue):
```

Figure 3 After editing the contact, the program will prompt the user if they want to continue editing contacts

- Sort

Additional feature of **sorting the contact list**. To accomplish this, we implemented the **Merge Sort algorithm**.

It is used to sort the contact list based on different criteria:

-Original contact list (-Name)

```
==========================================================
Contact 1
        Name: eve
        Phone number: 1234567890
        Email: eve@yahoo.jp


Contact 2
        Name: ibuki furusho
        Phone number: 0138290929
        Email: ramnjn0112mk@gmail.com


Contact 3
        Name: kenshi
        Phone number: 0123456789
        Email: kenshi@outlook.com

==========================================================
```

Figure 4.1 : The contacts will be auto sorted by name when display

-Phone number

```
Enter the number of what you would like to do, (-1 to exit): 5
What would you like to sort by?:
[1] Name
[2] Phone Number
[3] Email
[4] Latest Added

Enter your sort choice: 2

==================================================
Contact 1
        Name: kenshi
        Phone number: 0123456789
        Email: kenshi@outlook.com


Contact 2
        Name: ibuki furusho
        Phone number: 0138290929
        Email: ramnjn0112mk@gmail.com


Contact 3
        Name: eve
        Phone number: 1234567890
        Email: eve@yahoo.jp

==================================================
```

Figure 4.2 : The contacts will be sorted by phone number by using merge sort

-Email

```
**************************************************
Enter the number of what you would like to do, (-1 to exit): 5
What would you like to sort by?:
[1] Name
[2] Phone Number
[3] Email
[4] Latest Added

Enter your sort choice: 3

==================================================
Contact 1
        Name: eve
        Phone number: 1234567890
        Email: eve@yahoo.jp


Contact 2
        Name: kenshi
        Phone number: 0123456789
        Email: kenshi@outlook.com


Contact 3
        Name: ibuki furusho
        Phone number: 0138290929
        Email: ramnjn0112mk@gmail.com

==================================================

Contacts sorted successfully
```

Figure 4.3 : The contacts will be sorted by email

-Latest added

To enable sorting by the "latest added" order, we introduced an additional member in the contact structure. This member assigns an **index** to each contact whenever it is added. The index allows us to sort contacts in the order in which they were added, preserving their original sequence when required.

```
**************************************************
Enter the number of what you would like to do, (-1 to exit): 5
What would you like to sort by?:
[1] Name
[2] Phone Number
[3] Email
[4] Latest Added

Enter your sort choice: 4

==================================================
Contact 1
        Name: ibuki furusho
        Phone number: 0138290929
        Email: ramnjn0112mk@gmail.com


Contact 2
        Name: kenshi
        Phone number: 0123456789
        Email: kenshi@outlook.com


Contact 3
        Name: eve
        Phone number: 1234567890
        Email: eve@yahoo.jp

==================================================

Contacts sorted successfully
```

Figure 4.4 : The contacts will be sorted by index that indicates the added order

- Search (autocompletion)

  All search feature support multiple search - if there are two or more matching results, it will display all the matching results

-Name

```
Enter the number of what you would like to do, (-1 to exit): 6
What would you like to search by?:
[1] Name
[2] Phone Number
[3] Email

Enter your search choice: 1
Enter the search target: i

Contact found with keyword i:

Contact 1
        Name: ibuki furusho
        Phone number: 0138290929
        Email: ramnjn0112mk@gmail.com
```

Figure 5.1 : Search by name

-Phone number

```
Enter the number of what you would like to do, (-1 to exit): 6
What would you like to search by?:
[1] Name
[2] Phone Number
[3] Email

Enter your search choice: 2
Enter the search target: 0

Contact found with keyword 0:

Contact 1
        Name: kenshi
        Phone number: 0123456789
        Email: kenshi@outlook.com


Contact 2
        Name: ibuki furusho
        Phone number: 0138290929
        Email: ramnjn0112mk@gmail.com
```

Figure 5.2 : Accept multiple search

-Email address

```
Enter the number of what you would like to do, (-1 to exit): 6
What would you like to search by?:
[1] Name
[2] Phone Number
[3] Email

Enter your search choice: 3
Enter the search target: k

Contact found with keyword k:

Contact 1
        Name: ichikawa
        Phone number: 4536282940
        Email: kanagawa@gmail.com


Contact 2
        Name: kenshi
        Phone number: 0123456789
        Email: kenshi@outlook.com
```

Figure 5.3 : Search by email address with multiple matching results

- Save

When the user enters -1 (for exiting), it will automatically save the contacts information to a text file. It contains the total number of contacts, the shift number of Ceasar cipher for letters and numbers, and each name, phone number, email address and index for the latest added.

```c
void save_contacts(struct contact *pContact, int count, int letter_shift, int number_shift) {
    FILE *save_contact_list = fopen("contact_list.txt"/*change path accordingly*/, "w");
    if (save_contact_list == NULL) {
        printf("File could not be opened\n");
        return;
    }
    fprintf(save_contact_list, "%d\n", count);
    fprintf(save_contact_list, "%d\n", letter_shift);
    fprintf(save_contact_list, "%d\n", number_shift);
    for(int i = 0; i<count; i++) {
        fprintf(save_contact_list, "%s\n", pContact[i].name);
        fprintf(save_contact_list, "%s\n", pContact[i].phone);
        fprintf(save_contact_list, "%s\n", pContact[i].email);
        fprintf(save_contact_list, "%d\n", pContact[i].index);
    }
    fclose(save_contact_list);
}
```

Figure 6.1 : The code of saving function

```
Enter the number of what you would like to do, (-1 to exit): -1
Auto saving contacts...
Exiting Program...
```

Figure 6.2 : When -1 is entered, the program will exit with auto saving

- Load

  When the user has successfully logged into the program the contacts are loaded

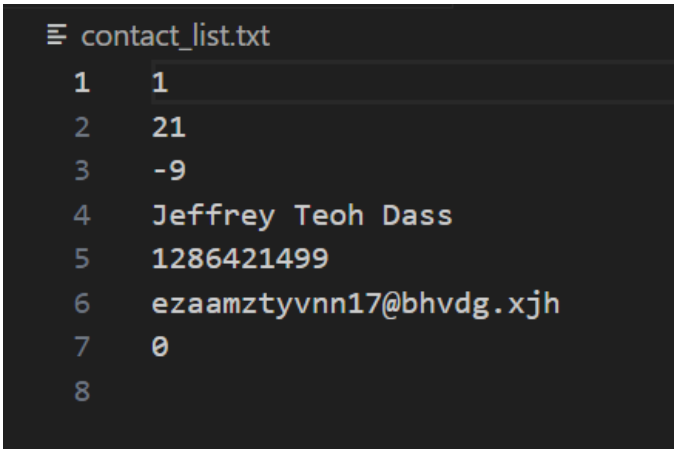


Figure 7.1 : Encrypted phone number (line 5) and email (line 6) in the contact_list text file

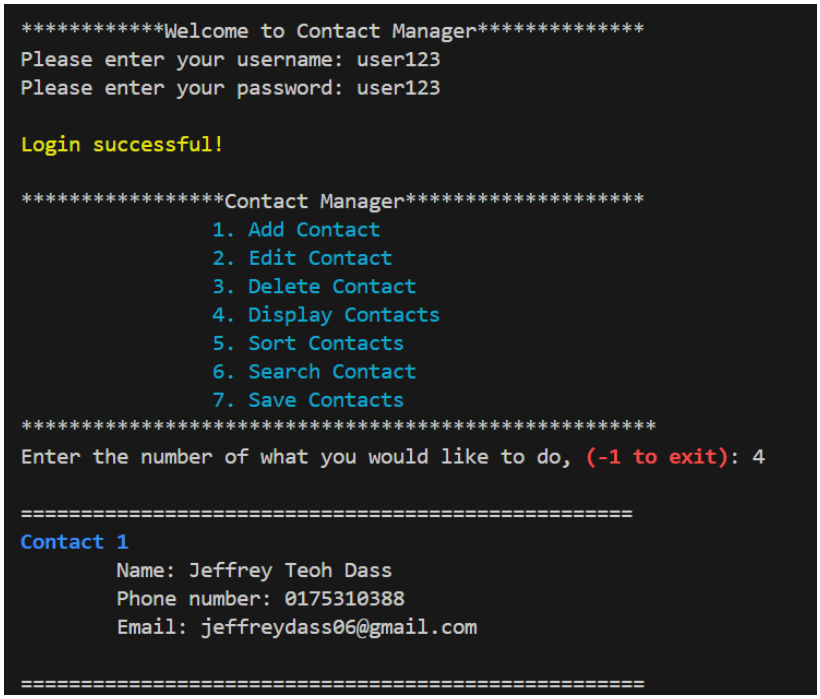


Figure 7.2 : Displayed contacts after loading and decryption

- Encryption and Decryption

  Every time the program saves to the text file whether it is auto-saving or through the save option. The program generates a random number and shifts all letters or numbers by that number before printing it into the text file

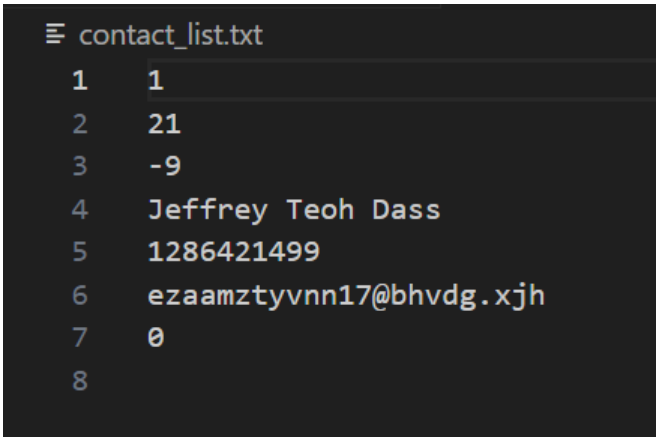


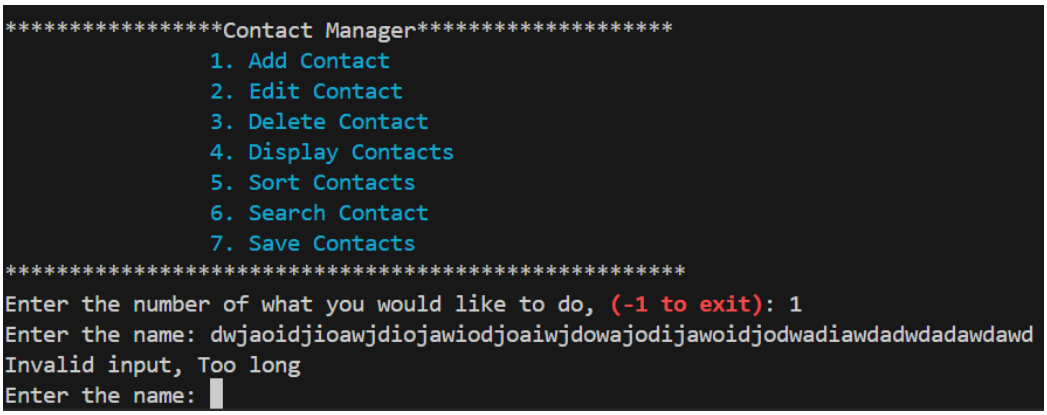
Figure 8: Line 1 is the contact count

Line 2 is the shift number of Ceasar cipher for letters

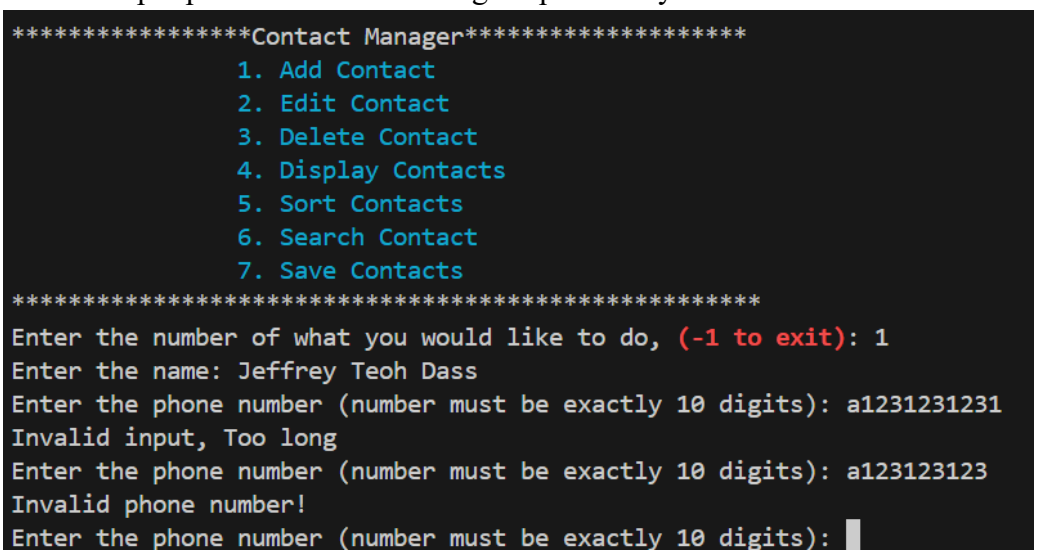
Line 3 is the shift number of Ceasar cipher for digits

Line 7 is the index to keep track of the latest added contacts

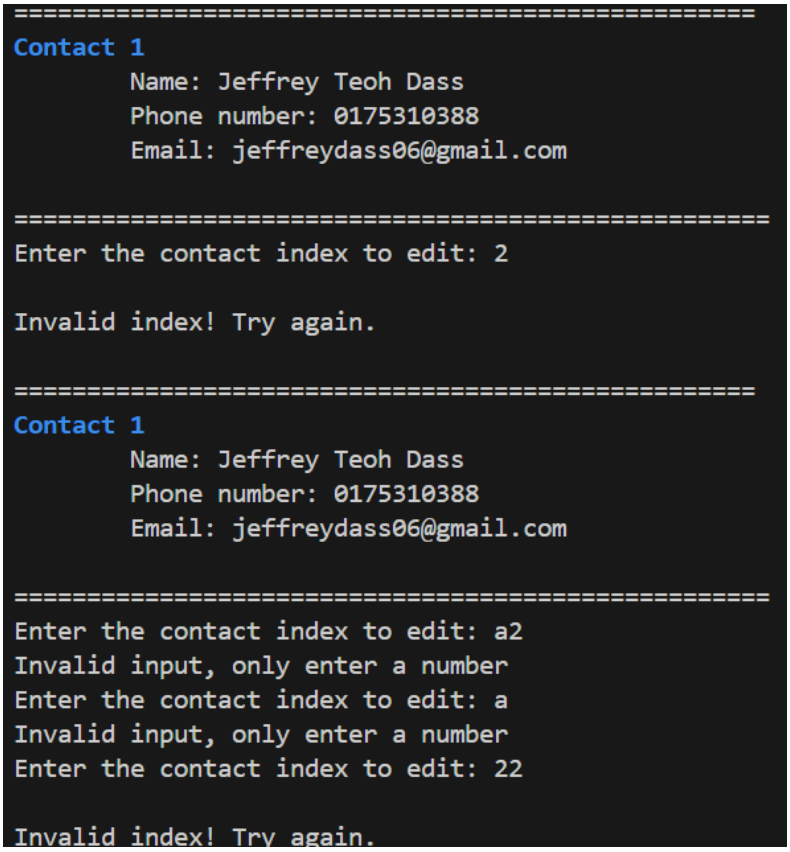
- Data validation
  - Size of array validation
    Ensures input does not overflow



  - Phone number validation
    Ensures input phone number is 10 digits specifically



- Index validation
  Ensures input index is within range and a valid input



5.0 Contact

Provide your contact information or how to reach you for questions or feedback.

- Your Name: Jeffrey Teoh Dass
- Your Email: jeffreydass06@gmail.com