



THE HONG KONG  
POLYTECHNIC UNIVERSITY  
香港理工大學

---

# **C++ Application Development Assignment – 6**

## **Phone Organizer**

### **Report**

Team: 15

Chan Shui Fung [15067344d]

Chui Tsz Lun [15070993d]

Choy Wing Shan [15081982d]

Date: 1/12/2015

Tutor in charge: Dr. Pauli Lai

# **Content**

1. Abstract
2. Introduction
3. Methodology
  - i. Division of work
  - ii. Schedule of developing the project
4. The structure of developed program
  - i. Specification
  - ii. Description
  - iii. The flow of execution
5. Problems encountered and ways to solve
6. Testing of the program
  - i. Process of testing
7. Results
8. Conclusion and further development

## **Abstract**

This assignment is about developing an application which can organize phone entries in a contact book which allow user to entry and store information of name, nickname, phone number, email address, last-call date and time. Moreover, users are allowed to store the entries under one of three groups specified by the user: Family, Friend or Junk. Also, the information of phone entries under a user-given group can be displayed or removed upon the user's request. The application requires user login so that the contact book would not be allowed to use unless username with corresponding password is provided.

## **Introduction**

A phone organizer is an application that can organize phone entries in a contact book which help users to store the phone entries in a more systematic way. The contact book would consist of information of contacts that perform under groups, functions for entry and remove contacts and a login system.

Users could input and save Information consist of name, nickname, phone number, email address, last-call date and time. Information should be stored in the contact book after users input them. A class *pEntry* is required to realize for holding the information.

The phone entries are stored under three groups: Family, Friend and Junk. The user can specify one of the three groups to store entries. A class *pContactB* is required to realize for holding a number of *pEntry* under three groups. The groups can be realized as an array of *pEntry* objects. And the size of each array is assigned to 100.

Users can request the application to display the information of entries in one specified group. A function `display()` is used to display the contact information under one of the three groups specified by the user. Under `display()`, a for-loop is used to show all the phone entry.

Users can request the application to display the number of valid elements in the array. A variable which store the number of valid elements of groups can is kept inside object *pContactB*. The size of each group is assigned to 100. If a contact is input by user, the number of valid elements should be minus one.

Users can remove phone entries under specified group. User is required to input the phone number that is chosen to delete. After user have chosen, the memory of that corresponding phone number will be released.

User need to login before start using the application. Username and password is required to type in before login. The program would read data from a file which is

used to store username and password. If the username is new to the program, new password would be required to input and they would be saved in the file storing username and password. If the user input correct username but fails to provide the correct password in three consecutive trials, the application will end.

The content of contact book should be saved into a file. The user should be able to read the content of current contact book from the file and store the information into three group arrays. Constructor is called by the program and read the content of the file and store the data into three groups.

## **Methodology**

### **Division of work**

The assignment is divided into two parts.

Part 1: Building of the program.

Building of the programming divide into 3 parts.

- A) Building of the contact book including input of phone entries under groups, display of information in contact book, removals of phone entries.
- B) Building of login system for the application such that the user need to login before using the application. User name and password are needed for login.
- C) Building of member functions for saving the current contact book into a file or load the current contact book from the file.

Part 2: Writing of the report.

Writing of the project divide into 4 parts.

- A) Abstract and Introduction including the objectives of the application. Also, a brief account of the method need to be conclude.
- B) Methodology including division of work, schedule of developing the project, structure of project, problems encountered and testing of the program.
- C) Result including the result show on screen after running the program.
- D) Conclusion including experience gained in the project and further development of the project.

Division of work	
<p>Chan Shui Fung 15067344D</p>	<p>Part 1: Building of the program. A) Building of the contact book C) Building of member functions for saving</p> <p>Part 2: Writing of the report. B) Methodology including structure of project, problems encountered and testing of the program</p>
<p>Choy Wing Shan 15081982D</p>	<p>Part 2: Writing of the report. A) Abstract and Introduction B) Methodology including division of work, schedule of developing the project D) Conclusion including experience gained.</p>
<p>Chui Tsz Lun 15070993D</p>	<p>Part 1: Building of the program. B) Building of login system</p> <p>Part 2: Writing of the report. C) Result D) Conclusion including further development of the project</p>

### **Schedule of developing the project**

Schedule of developing the project	
5 <sup>th</sup> November, 15	First discussion on division of work.
5 <sup>th</sup> November, 15 ~ 18 <sup>th</sup> November, 15	Developing of the application and writing part of the report: Introduction and Methodology.
18 <sup>th</sup> November, 15	First in-group deadline on program writing and discussion on how to improve the program.
19 <sup>th</sup> November, 15 ~ 25 <sup>th</sup> November, 15	Writing the remaining part of the report: Result and Conclusion.
25 <sup>th</sup> November, 15	Second in-group deadline on report writing and discussion on how to improve the report. And the program and report should be finished.
1 <sup>st</sup> December, 15	Deadline of the assignment.

## **The structure of developed program**

To develop an object-oriented console application that can organize phone entries in a contact book, we use class to be a prototype that defines the variables and methods common to all objects of a certain kind. Class saves effort in developing the phone organizer as there may have a large number of phone entries in a contact book.

We designed a class *pEntry* holding the information of name, nickname, phone number, email address, last-call date and time. Every object of *pEntry* should be a phone entry and every phone entry have their own information. Thus, we designed some methods which are the member function of *pEntry* to organize the information of phone entry.

Also, we designed a class *pContactB* holding a number of *pEntry* objects under three groups: Family, Friend and Junk. Every group can be realized as an array of *pEntry* objects. Every object of *pContactB* should be a contact book and every contact book have their own group. Each group have a number of phone entry. Thus, we designed some methods which are the member function of *pContactB* to organize the information of contact book, save the content into a file and read a content from a file which contain the data of phone entries.

In addition, for the final design of the console application, we designed a function to provide a text-mode user interface. Hence, the user can repeatedly organize and display the content of the contact book until the user chooses to end the application. Furthermore, a function will follow the user's request to input the phone entry and store it into a *pContactB* object under one of the three groups specified by the user. The application can also display the information of phone entries under a user-given group and the status of the storage. An extra login function has been designed to provide a validation of user identity.

## **Specification**

### **Class *pEntry***

In *pEntry*, we defined some private member variables that store the information of a phone entry. Those private member variables can only be accessed by the member function of that class. It reduces the possibility of erroneous situations.

Variables name	Type	Description
itsName	Character array	Name of contact The array size is assigned to 80.
itsNickname	Character array	Nickname of contact The array size is assigned to 50.
itsPhoneNo	Character array	Phone number of contact The array size is assigned to 15. It is assigned to character array as calculation is unnecessary.
itsEmail	Character array	Email of contact The array size is assigned to 100.
itsLastCallDateAndTime	Integer	Last call date and time of contact

All of the character array are assigned to an acceptable size.

Moreover, we defined some public member function that input and retrieve the information of a contact. These function can be divided into two group: set function and get function.

#### Set function

Function name	Input parameter	Output parameter
setName	Character array	Nil
setNickname	Character array	Nil
setPhoneNumber	Character array	Nil
setEmail	Character array	Nil
setLastCallDateAndTime	Integer	Nil

### **Description**

setName will receive a user-given name of a phone entry and set the name to itsName.

setNickname will receive a user-given nickname of a phone entry and set the nickname to itsNickname.

setPhoneNumber will receive a user-given phone number of a phone entry and set the phone number to itsPhoneNo.

setEmail will receive a user-given email of a phone entry and set the email to itsEmail.

setLastCallDateAndTime will receive a user-given last call date and time of a phone entry and set the email to itsLastCallDateAndTime.

#### Get function

Function name	Input parameter	Output parameter
getName	Nil	Character array
getNickname	Nil	Character array
getPhoneNo	Nil	Character array
getEmail	Nil	Character array
getLastCallDateAndTime	Nil	Integer



## Description

getName will return the value of itsName.

getNickname will return the value of itsNickname.

getPhoneNo will return the value of itsPhoneNo.

getEmail will return the value of itsEmail.

getLastCallDateAndTime will return the value of itsLastCallDateAndTime.

All of the get function will never change the value of any member variable. It is desirable to declare them as constant member function. Then the compiler will automatically check if there is any inconsistency in the program and helps to debug the program.

pEntry
-char itsName[80] -char itsNickname[50] -char itsPhoneNo[15] -char itsEmail[100] -int itsLastCallDateAndTime
+setName(name: const char): void +setNickname(nickname: const char): void +setPhoneNumber(phoneNo: const char): void +setEmail(email: const char): void +setLastCallDateAndTime(dateAndTime: int): void +getName(): itsName +getNickname(): itsNickName +getPhoneNo(): itsPhoneNo +getEmail(): itsEmail +getLastCallDateAndTime(): itsLastCallDateAndTime

## **Class *pContactB***

In *pContactB*, we defined some private member variables. The variable store the number of valid elements of each array and pointer of *pEntry* object.

Variables name	Type	Description
Family	Pointer	A pointer that point to a member array of <i>pEntry</i> object under group Family
Friend	Pointer	A pointer that point to a member array of <i>pEntry</i> object under group Friend
Junk	Pointer	A pointer that point to a member array of <i>pEntry</i> object under group Junk
familyA	Integer	The number of valid elements of each array in group Family
friendA	Integer	The number of valid elements of each array in group Friend
junkA	Integer	The number of valid elements of each array in group Junk

Moreover, we defined some public member function that organize the information of contact book.

Function name	Input parameter	Output parameter
pContactB <Constructor>	Nil	Nil
~pContactB <Destructor>	Nil	Nil
inputFamily	Nil	Nil
inputFriend	Nil	Nil
inputJunk	Nil	Nil
display	Integer	Boolean
getFamilyA	Nil	Integer
getFriendA	Nil	Integer
getJunkA	Nil	Integer
deleteF	Integer	Integer

## **Description**

**pContactB (Constructor)** will create three array of object of *pEntry* in the heap which pointed by Family, Friend and Junk first. A contact book must create a number of *pEntry* object to store the phone entry. Thus, the array of object of *pEntry* is declared inside the constructor. Each size of an array of *pEntry* object is assigned to 100 in order to ensure the application provide adequate phone entry storage. As a computer may not have enough memory to store in total 300 objects in the stack, our group decided to create the objects in the heap. The memory in free store should be large enough to store those objects.

Then, the function read the content of the file "contactB.txt" and store the information into three member arrays of *pEntry* object. Data input from a file should be done at the very beginning when the object of pContactB has been declared. Hence, the input process should be done in the constructor so that he user could organize the phone entries which retrieve from a data file via the application.

**~pContactB (Destructor)** will save the content of the current contact book into a file "contactB.txt", release the memory which allocated by *pEntry* object to the system and assign the pointer with 0 after delete. When the user request to end the program, all of the alteration of phone entries should be save into the data file for further usage. Hence, our group chooses to save all of the content of the current contact book into the data file in the destructor. To avoid the memory leaks, the memory which allocated by *pEntry* object should release to the system. Also, to ensure one will not use the deleted heap memory again, assign the pointer with 0 after delete.

To achieve the requirement of the input of phone entry and store it into a pContactB object under one of the three groups specified by user, three input member function in pContactB are defined for insertion of each group.

**inputFamily** will ask the user to input the information of a phone entry and store it into a pContactB object under group Family.

**inputFriend** will ask the user to input the information of a phone entry and store it

into a **pContactB** object under group Friend.

**inputJunk** will ask the user to input the information of a phone entry and store it into a **pContactB** object under group Junk.

The number of phone entry will be limited by the variable which store the number of valid elements as the number of phone entry should not exceed the array size. It prevents result unpredictable. After an insertion of phone entry, the number of valid elements under a specified group should be increment by 1 to ensure the data integrity.

**display** will display the contact information under one of the three groups which specified by the user. It receive an integer as input parameter which indicate the user-given group. If the integer is 1, 2 or 3, it display the content of group Family, Friend or Junk respectively. The number of phone entry display will be limited by the variable which store the number of valid elements. It prevents to show the phone entry which is not initialize.

In order to display the storage status of the phone entry, the variable that stores the number of valid elements of each group should be able to retrieve. These functions will never change the value of any member variable so we declared them as constant member function.

**getFamilyA** will return the value of FamilyA.

**getFriendA** will return the value of FriendA.

**getJunkA** will return the value of JunkA.

**deleteF** will remove the phone entry which match a user-input phone number under a user-given group. It receive an integer as input parameter which indicate the user-given group. If the integer is 0, it means there is no phone entry under user-given group, then the function ends. If the integer is 1, 2 or 3, it display the content of group Family, Friend or Junk respectively and remove a phone entry which match the user-given phone number. Our group considered a situation that if the user do not want to remove any phone entry, he/she could input 'q' to quit the function. After a specified phone entry is confirmed to remove, all the elements in the array that follow that entry will be moved forward and the last phone entry will be set as null. It assure the valid entries are stored consecutively.

pContactB
-int familyA = 100 -int friendA = 100 -int junkA = 100 -pEntry * Family -pEntry * Friend -pEntry * Junk
«constructor»+pContactB() «destructor»~pContactB() +inputFamily(): void +inputFriend(): void +inputJunk(): void +display(gchoose: int): bool +getFamilyA(): FamilyA +getFriendA(): FriendA +getJunkA(): JunkA +deleteF(group: int): int

## Console Application

The final design of the console application provide a text-mode user interface to organize and display the content of the contact book repeatedly. An extra login function has been designed to provide a validation of user identity.

Function name	Input parameter	Output parameter
gChoose	Nil	Integer
menu	Object	Boolean
login	Nil	Boolean
main	Nil	Integer

**gChoose** will show a menu and ask for user to decide the type of group. It will return an integer to indicate the decision of user for further use. The returned value determine the user-given group. It return 1, 2 or 3 if the user chooses group Family, Friend or Junk respectively. It return 4 if the user wants to quit the group selection. It return 0 if the user input invalid value such as alphabet character or any number larger than 4. The function is used to assist the input, delete and display member function in *pContactB* as these function request an input parameter which is the user-given group.

**menu** will receive the object pointed by a pointer which is passed by caller. The function show a user menu and follow the user's request. It will return a boolean that indicate the end of the function and also the end of the program. The function will ask the user to input an option which is used to indicate the task. After the user decide an option, the function will run a specific task.

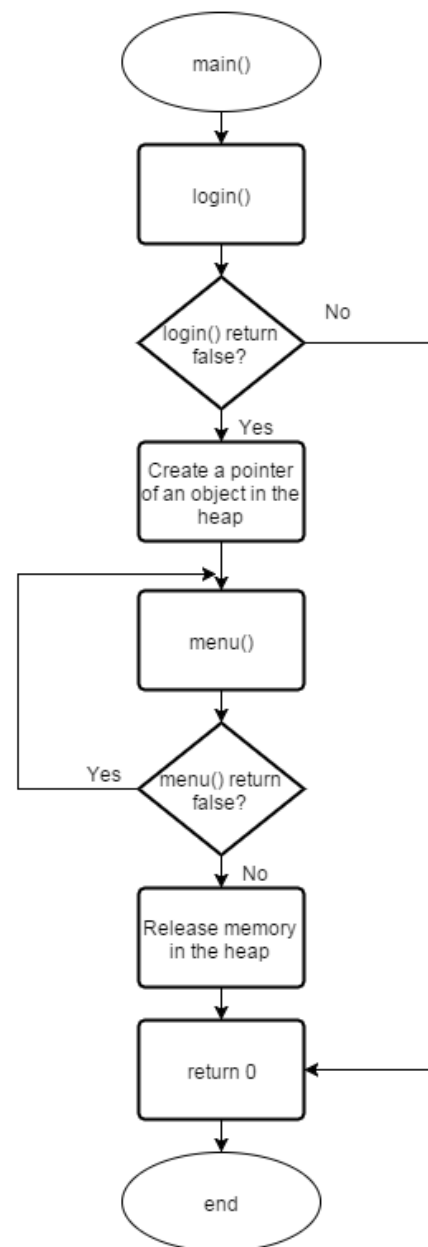
**login** will ask the user to input a user name first and check a file that stores username-password pairs. If the username is new, the application will ask for a password from the user and store it into that file. If the username can be found in that file, the function will ask the user to input a password. Application will start only when the password is correct. The application will end if the user fails to provide the

correct password in three consecutive trials.

**main** is used to indicate the program flow. It gives the skeleton of program. The main function will prompt the user to login. If login successful, it will create a pointer of an object of a class *pContactB* in the heap. Then, it repeatedly show a menu until the user request to end the program. At last, it release the memory which allocated by object of *pContactB* to the system.

## The flow of execution

The main flow of execution of the application is indicated by the main function. Main function gives the skeleton of program. It calls login() and menu() which provide a login function and a text-mode interface. The main function will ask the user to login. If login() return true, it will create a pointer of an object of a class *pContactB* in the heap. Else, the program ends. Then, it repeatedly show a menu until menu() return true. At last, it release the memory in the heap and return 0.

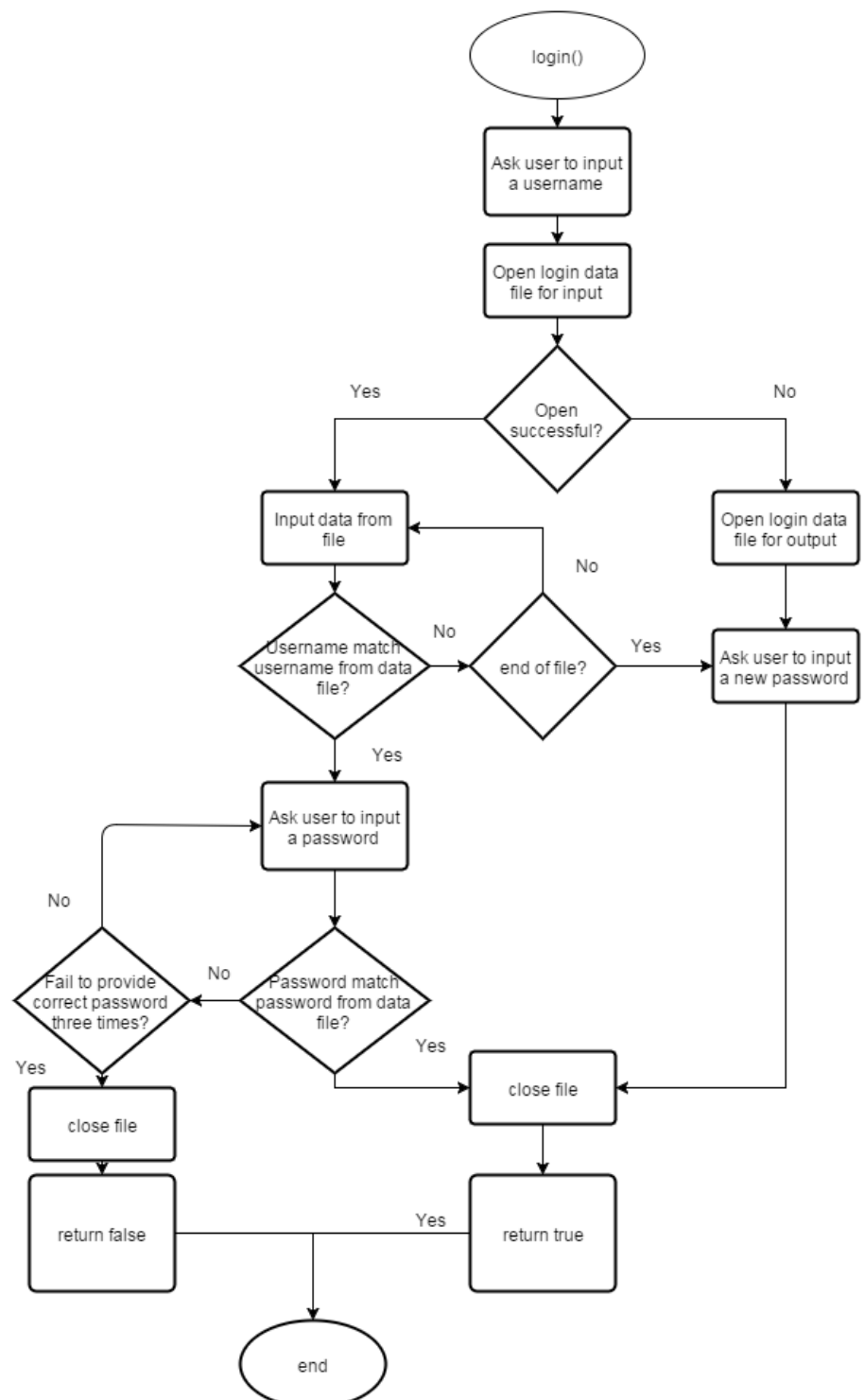


- Flowchart of main()

Login function is called by main function and the flow of login() is important to the whole application also. The function ask user to input a username first and open login data file for input data.

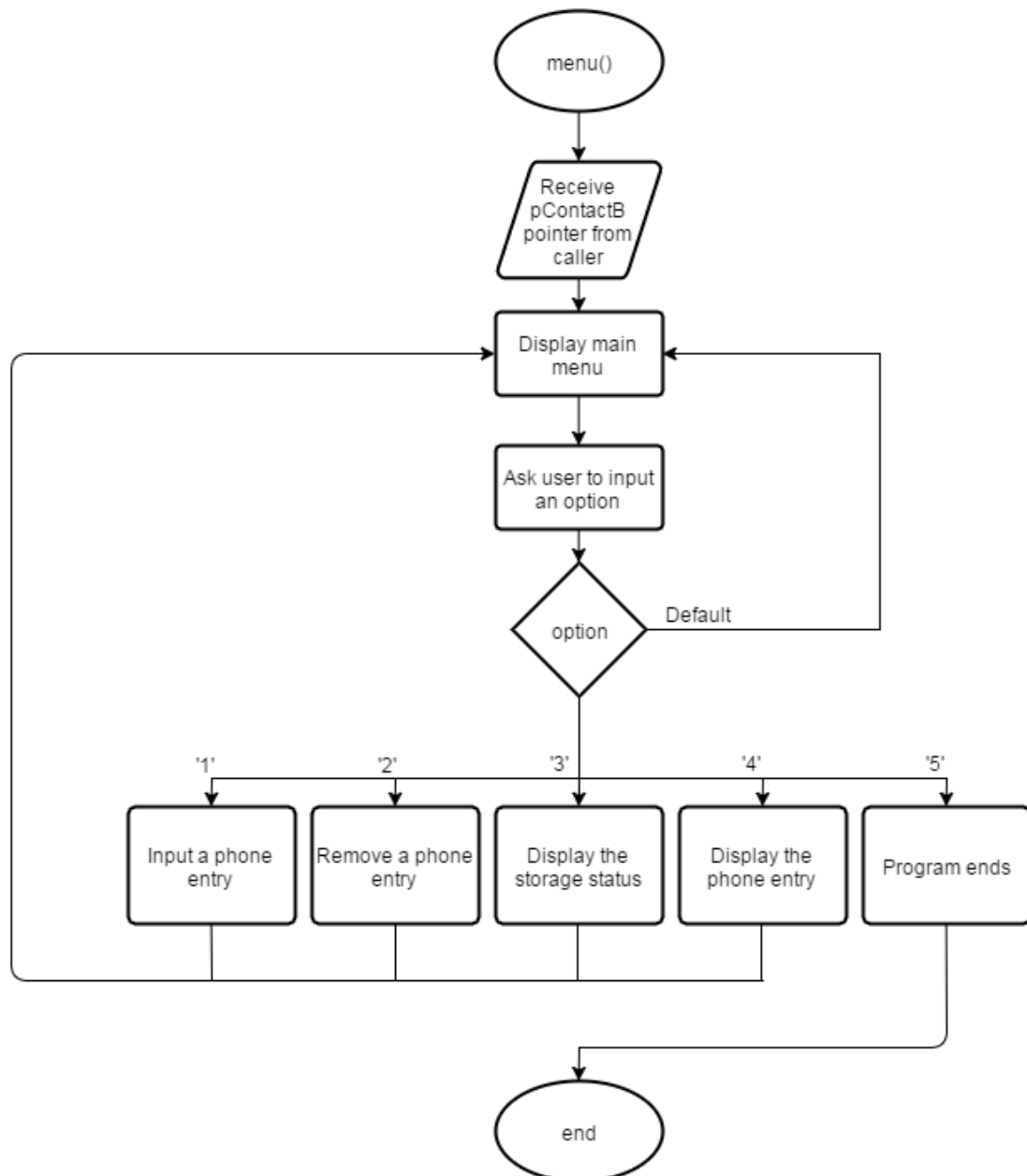
If it is successful, input data from the file and check if there is any username that match the user-given user name. If no username match, the function ask user to input a new password and the password will store into the file in append mode. The function return true and ends. If there is any username that match the user-given user name. The function will ask user to input a password and check the password. If the password is correct, function return true. If not, input again until the user fail to provide correct password three times. The function will return false in this situation.

If the login data file is fail to open, the function will treat it as a new user and open a new login data file. The function ask user to input a new password and the password will store into the file.



- Flowchart of login()

Menu function provide a text-mode interface for user so that they can repeatedly organize and display the content of the contact book until the user chooses to end the application. The menu helps the end user to control the flow of the application upon the user's request. The function will receive *pContactB* pointer from caller first and display a main menu which instruct the user to input his/her option. User may input '1' to input a phone entry under user-given group, '2' to remove a phone entry under user-given group, '3' to check the number of valid elements, '4' to display the phone entry under user-given group and '5' to quit. It will display an error message when the user input an invalid input. It will only ends if the user choose to quit.



- Flowchart of menu()



## **Problems encountered and ways to solve**

1. The application consist of few parts such as the class definition or the console application. Hence, we divide the practical work of programming to three modules and handled by two group members. We develop the program modules independently and try to integrate them together. However, the program contains many errors and the modules are incompatible. The name of variables are inconsistent and the functions are not well defined. Although we correct the errors owing to incorrect use of language syntax or unresolved references, the solution do not fulfill the requirement and some run-time error occur. The failure of the program is a big problem of our team.

To solve this problem, our group decided to start afresh. We realized that we forgot an important step in modular program design which is the definition of specifications of each module. The specifications include the function, the calling methods, the parameters returned, the variable and the function name. Moreover, based on the requirements, we should analysis it and design the program flow of the application. It enhance the integrity of the program and reduce the opportunity of inconsistency. Building a flow chart help us to develop a structural program and the all modules are correct irrespective to the difference in the code if the specifications are met.

We build a UML class diagram to define the member function and member variable. The diagram helps us to have a consistent reference. Also, we design the program flowchart to indicate the flow of execution. It assure that the program modules will be developed under the same structure and met the specifications.

2. The program size is not small, it is not easy to debug such a program. Indeed, after we integrated our program modules, we found that the program has many run- time errors. Comparing with compile-time errors, run- time errors are more difficult to locate and correct. It may consists some logic errors which is very difficult to locate. These run-time errors lead to incorrect results and even system failure caused by infinite loop. As the program is integrated, it consists some separate static library and linked into the console application. The program was not in module at that time so it is hard to debug.

To solve this problem, our group decided to divide the program again. We tried to build the class as a console application and add stubs for testing. We added a simple main() to test any anomaly in both class. By using run time debugger, it was easier to trace the program flow in modules approach. Also, we were achieved the location of errors by divide and conquer .Hence, we debugged the program rapidly and it saved time for the whole process of debugging.

3. The ability of team mate are different. Not every team members are good at programming. The work division among team members should consider deeply in order to enhance the efficiency of the work. However, all of us are Year 1 students and we know each other just few months. We did not know the ability of each other.

Thus, we had our first meeting at 5<sup>th</sup> November, 15. We discussed the division of work and we know the ability of team mate.

4. In the week of the first deadline, we had three quiz or test of other subject. We were difficult to finish the work on time as we did not have enough time to do the work. Thus, we decided to setup a second deadline to ease the workload of us.

## Testing of the program

To validate the program, we applied some sample data to the program.

The number of sample data is 30 phone entries: 10 of them under Family, 10 of them under Friend and 10 of them under Junk.

	A	B	C	D	E	F
1	Group	Name	Nickname	Phone no	Email	Last call date (MMDDHHSS)
2	0	Chan_Ho	Assho	61234567	61234567@gmail.com	1020304
3	0	Yeung_Yi	Fakenose	53995369	53995369@gmail.com	11220304
4	0	Shun_Lai_Heung	Heung	96838759	96838759@gmail.com	1262310
5	0	Chan_Kaki	Kaki	92011811	92011811@gmail.com	12121212
6	0	Chan_sze_sze	Sze	64806099	64806099@gmail.com	10212110
7	0	Wong_Yan	Yan	69327523	69327523@gmail.com	4040112
8	0	Kwong_king_man	Kingman	96627435	96627435@gmail.com	4041201
9	0	Au_you	Wai	93497790	93497790@gmail.com	12131415
10	0	Wong_kei	Kei	95412919	95412919@gmail.com	12141516
11	0	Lei_ka_shing	Shing	91687130	91687130@gmail.com	10022308
12	1	Ho_Ho	Ho	96635501	96635501@gmail.com	10081654
13	1	Ho_Tung	Tung	65710577	65710577@gmail.com	12190136
14	1	Ko_mut_chun	boss	56004556	56004556@gmail.com	6051114
15	1	Choy_chi_Ching	Ching	98432644	98432644@gmail.com	8050505
16	1	Lam_chin	Chin	54884721	54884721@gmail.com	5080609
17	1	Wu_hin	Ivan	60842397	60842397@gmail.com	9251526
18	1	Denise_Wong	Denise	67695400	67695400@gmail.com	12151656
19	1	Kenneth_Wu	Kenneth	54451194	54451194@gmail.com	4031539
20	1	Jack_Lam	Jack	68332233	68332233@gmail.com	3022359
21	1	Jessica_Chan	Jc	66862316	66862316@gmail.com	9011230
22	2	Bank_of_China	China_bank	26972179	26972179@gmail.com	6260936
23	2	HSBC	HSBC_bank	25242900	25242900@gmail.com	12121230
24	2	Heng_seng_bank	Hsb	26085532	26085532@gmail.com	12301836
25	2	Standard_charted	Sc	29223352	29223352@gmail.com	12121315
26	2	cc_fitness	Cc	23325501	23325501@gmail.com	6261338
27	2	Cha_ka_chak	Ka_chak	36058323	36058323@gmail.com	11242348
28	2	CY_Leung	689	31331780	31331780@gmail.com	9090809
29	2	Skin_beauty	Skin	34695987	34695987@gmail.com	6080915
30	2	DBS	Dbs_bank	37695072	37695072@gmail.com	11241936
31	2	3HK	3HK	39009793	39009793@gmail.com	12152337

The validation will use all data for testing to confirm that the solution is correct and fulfill all of the requirement. The application requirement mainly consist of five part:

- 1) Login the application and check the user name and password.
- 2) Input the phone entries under user-given group.
- 3) Remove the phone entries under user-given group.
- 4) Display the phone entries under user-given group.
- 5) Save the content of current contact book into a file and read the data from a data file and store it.

### **Process of testing**

First, we will login to the application using a test user name and password. We will observe any run-time errors or anomaly in the application. The test validates the login function of the application which confirm the solution is correct.

Second, we input those sample data as phone entries into the application. We will check if any phone entries cannot be added successfully. It validate that the input function of the application.

Third, we will try to remove one of the record and see what will happen if we input an incorrect phone number. We will check the success of deletion and any anomaly in the application. The test validates the remove function of the application.

Fourth, we will select an option to display the phone entries under different group. We will check the phone entries if the application display any incorrect record. The test validates the display function of the application.

Finally, we will select an option to end the program and open the file which contain the username-password pair and the information of phone entries. We will check if there is any inconsistency in the data file. Then, we will run the program again and check whether the application has read the content of the data file. The test validates the save and read file function of the application.

In addition, we considered a situation that there is no phone entries inside the application. Also, we have considered if the application is full of phone entries. The application should show in different way under these two different situation. For example, if there is no phone entry, remove and display should not be done. Moreover, if there is full of entries, the input should not be done.

## Results

To be validate the program, we will divide this part into two sections. A normal input and an input that should not be done by the users.

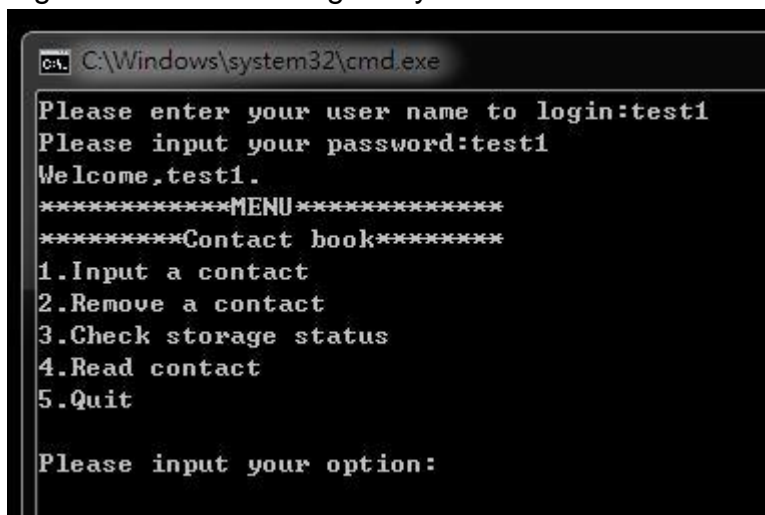
When the program runs, the program will ask the user to input their username. When the username is new to the program, it should ask to input a new password to create a new account for login in future. (Figure 1)

When the user login again, the program will only ask the password instead of creating a new account for the same username. (Figure 2)



```
CA: C:\Windows\system32\cmd.exe
Please enter your user name to login:test1
Welcome, new user!
Please input your new password:test1
New account has been created.
Welcome,test1.
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit
Please input your option:
```

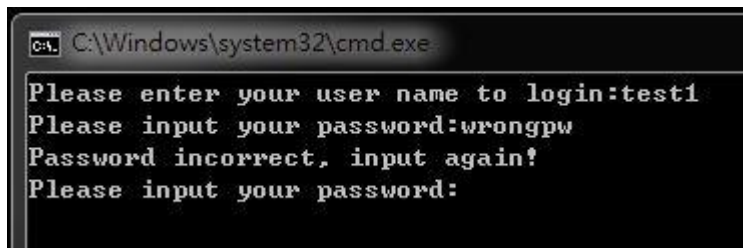
Figure 1: New User Login Layout



```
CA: C:\Windows\system32\cmd.exe
Please enter your user name to login:test1
Please input your password:test1
Welcome,test1.
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit
Please input your option:
```

Figure 2: User re-login layout

When the user input a wrong password, a message will alert the user the password is wrong and let the user input again. (Figure 3)

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The text inside the window reads: 'Please enter your user name to login:test1', 'Please input your password:wrongpw', 'Password incorrect, input again!', and 'Please input your password:'.

```
C:\Windows\system32\cmd.exe
Please enter your user name to login:test1
Please input your password:wrongpw
Password incorrect, input again!
Please input your password:
```

(Figure 3)

But when the user type in wrong password 3 in a row, the program will ends. (Figure 4)

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The text inside the window reads: 'Please enter your user name to login:test1', 'Please input your password:123456', 'Password incorrect, input again!', 'Please input your password:abcdefg', 'Password incorrect, input again!', 'Please input your password:abc123', 'Password incorrect, input again!', 'Fails to provide correct password. Program ends.', and '請按任意鍵繼續 . . .'.

```
C:\Windows\system32\cmd.exe
Please enter your user name to login:test1
Please input your password:123456
Password incorrect, input again!
Please input your password:abcdefg
Password incorrect, input again!
Please input your password:abc123
Password incorrect, input again!
Fails to provide correct password. Program ends.
請按任意鍵繼續 . . .
```

(Figure 4)

After the user logged into the program, a menu will show and let the user to choose from 5 options: Input, Remove a contact, Check status, read a contact and quit the program.

When the user choose option 1, the program will ask to choose one of three group that will store the new contact. Hence, the user should input the name, nickname, phone number, email, and last-call date one by one. Finally, all the data will store into a file when the program ends. (Figure 5)

```
C:\Windows\system32\cmd.exe

Please enter your user name to login:test1
Please input your password:test1
Welcome,test1.
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:1
*****
*****GROUP*****
1.Family
2.Friend
3.Junk
4.Quit

Please input your decision:1
*****
Please use underscore(_) to represent a space!!
Name:Chan_Ho
Nickname:Aho
Phone Number:61234567
Email:61234567@gmail.com
Last-call date and time(MMDDHHSS):1020304
Record 1 in Family has been created.
Available elements in Family :99
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:
```

(Figure 5)

When the user try to delete a data, the user are required to type in the phone number to delete a record. (Figure 6)



```
C:\Windows\system32\cmd.exe
Last-call date and time(MMDDHHSS):4041201
*****8*****
Name:Au_you
Nickname:Wai
Phone Number:93497790
Email:93497790@gmail.com
Last-call date and time(MMDDHHSS):12131415
*****9*****
Name:Wong_kei
Nickname:Kei
Phone Number:95412919
Email:95412919@gmail.com
Last-call date and time(MMDDHHSS):12141516
*****10*****
Name:Lei_ka_shing
Nickname:Shing
Phone Number:91687130
Email:91687130@gmail.com
Last-call date and time(MMDDHHSS):10022308
*****
Please input the phone number of record which you would like to delete or 'q' to
quit:61234567
Record 1 has been deleted.
Available elements in Family :91
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:
```

(Figure 6)

When the user type a wrong number, the program will not execute anything and go back to the menu, i.e. the delete function will only run when the user type in a correct phone number and there is at least one record inside the corresponding group.

(Figure 7) (Figure 8)



```
C:\Windows\system32\cmd.exe
Last-call date and time(MMDDHHSS):12131415
*****8*****
Name:Wong_kei
Nickname:Kei
Phone Number:95412919
Email:95412919@gmail.com
Last-call date and time(MMDDHHSS):12141516
*****9*****
Name:Lei_ka_shing
Nickname:Shing
Phone Number:91687130
Email:91687130@gmail.com
Last-call date and time(MMDDHHSS):10022308
*****
Please input the phone number of record which you would like to delete or 'q' to
quit:999
The phone number does not have any phone entry!
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit
Please input your option:
```

(Figure 7)

```
C:\WINDOWS\system32\cmd.exe
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit
Please input your option:2
*****
*****GROUP*****
1.Family
2.Friend
3.Junk
4.Quit
Please input your decision:1
*****
No record is found.
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit
Please input your option:
```

(Figure 8)

For the storage parts, when there is no any record stored inside the program, the storage status of each group should be 100. And after a new contact is created, the storage will be decreased by one to 9. (Figure 9) (Figure 10)

```
C:\Windows\system32\cmd.exe

*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:3
*****
The number of valid element of group Family:91
The number of valid element of group Friend:90
The number of valid element of group Junk:90
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:
```

(Figure 9)

```
C:\Users\Eddie\Desktop\Assignment_draft 7\Assignment_draft 7\Application\Debug\Application.exe

Last-call date and time(MMDDHHSS):000999
Record 1 in Family has been created.
Available elements in Family :99
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:3
*****
The number of valid element of group Family:99
The number of valid element of group Friend:100
The number of valid element of group Junk:100
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:
```

(Figure 10)

For the reading function, the function only will run when there is one or more data store in the corresponding group, otherwise “No record is found” will be shown. If there is a record found, all of the data related will be show with a number that state the number of record that the user type in. (Figure 11) (Figure 12)

```
C:\WINDOWS\system32\cmd.exe
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:4
*****GROUP*****
1.Family
2.Friend
3.Junk
4.Quit

Please input your decision:1
*****MENU*****
*****Contact book*****
1.Input a contact
2.Remove a contact
3.Check storage status
4.Read contact
5.Quit

Please input your option:
```

(Figure 11)

```
*****1*****
Name:Yeung_Yi
Nickname:Fakenose
Phone Number:53995369
Email:53995369@gmail.com
Last-call date and time(MMDDHHSS):11220304
*****2*****
Name:Shun_Lai_Heung
Nickname:Heung
Phone Number:96838759
Email:96838759@gmail.com
Last-call date and time(MMDDHHSS):1262310
*****3*****
Name:Chan_Kaki
Nickname:Kaki
Phone Number:92011811
Email:92011811@gmail.com
Last-call date and time(MMDDHHSS):12121212
*****4*****
Name:Chan_sze_sze
Nickname:Sze
Phone Number:64806099
Email:64806099@gmail.com
Last-call date and time(MMDDHHSS):10212110
*****5*****
Name:Wong_Yan
Nickname:Yan
Phone Number:69327523
Email:69327523@gmail.com
Last-call date and time(MMDDHHSS):4040112
*****6*****
Name:Kwong_king_man
Nickname:Kingman
Phone Number:96627435
Email:96627435@gmail.com
Last-call date and time(MMDDHHSS):4041201
*****7*****
Name:Au_you
Nickname:Wai
Phone Number:93497790
Email:93497790@gmail.com
Last-call date and time(MMDDHHSS):12131415
*****8*****
Name:Wong_kei
Nickname:Kei
Phone Number:95412919
Email:95412919@gmail.com
Last-call date and time(MMDDHHSS):12141516
*****9*****
Name:Lei_ka_shing
Nickname:Shing
Phone Number:91687130
Email:91687130@gmail.com
Last-call date and time(MMDDHHSS):10022308
```

(Figure 12)

There will be two file that store the login data and the contact data correspondingly output when the program ends. (Figure 13)

The screenshot shows two Notepad windows. The top window, titled 'logindata - 記事本', contains the text 'test1 test1'. The bottom window, titled 'contactB - 記事本', contains a list of contact information with columns for an index, name, phone number, email, and another number.

Index	Name	Phone Number	Email	Another Number
0	Chan_Ho Assho	61234567	61234567@gmail.com	1020304
0	Yeung_Yi Fakenose	53995369	53995369@gmail.com	11220304
0	Shun_Lai_Heung	Heung 96838759	96838759@gmail.com	1262310
0	Chan_Kaki	Kaki 92011811	92011811@gmail.com	12121212
0	Chan_sze_sze	Sze 64806099	64806099@gmail.com	10212110
0	Wong_Yan	Yan 69327523	69327523@gmail.com	4040112
0	Kwong_king_man	Kingman 96627435	96627435@gmail.com	4041201
0	Au_you Wai	93497790	93497790@gmail.com	12131415
0	Wong_kei	Kei 95412919	95412919@gmail.com	12141516
0	Lei_ka_shing	Shing 91687130	91687130@gmail.com	10022308
1	Ho_Ho Ho	96635501	96635501@gmail.com	10081654
1	Ho_Tung Tung	65710577	65710577@gmail.com	12190136
1	Ko_mut_chun	boss 56004556	56004556@gmail.com	6051114
1	Choy_chi_Ching	Ching 98432644	98432644@gmail.com	8050505
1	Lam_chin	Chin 54884721	54884721@gmail.com	5080609
1	Wu_hin Ivan	60842397	60842397@gmail.com	9251526
1	Denise_Wong	Denise 67695400	67695400@gmail.com	12151656
1	Kenneth_Wu	Kenneth 54451194	54451194@gmail.com	4031539
1	Jack_Lam	Jack 68332233	68332233@gmail.com	3022359
1	Jessica_Chan	Jc 66862316	66862316@gmail.com	9011230
2	Bank_of_China	China_bank 26972179	26972179@gmail.com	6260936
2	HSBC	HSBC_bank 25242900	25242900@gmail.com	12121230
2	Heng_seng_bank	Hsb 26085532	26085532@gmail.com	12301836
2	Standard_chartered	Sc 29223352	29223352@gmail.com	12121315
2	cc_fitness	Cc 23325501	23325501@gmail.com	6261338
2	Cha_ka_chak	Ka_chak 36058323	36058323@gmail.com	11242348
2	CY_Leung	689 31331780	31331780@gmail.com	9090809
2	Skin_beauty	Skin 34695987	34695987@gmail.com	6080915
2	DBS	Dbbs_bank 37695072	37695072@gmail.com	11241936
2	3HK	3HK 39009793	39009793@gmail.com	12152337

(Figure 13)

## **Conclusion and further development**

In conclusion, we have a great chance to having an experience of the real situation of the development of an application which is not able to complete since it can't be a one-man job.

We needed to divide the program into multiple parts which can be specialize by our skills. In our group, some of us are having a great performance in pointers and array, some of us are experienced in file I/O and some of us are good at logical thinking that can plan the flow of the program and avoid causing bugs when different parts of the program combine together.

Moreover, we have experienced that we always need to plan all the event ahead of any schedule since it is always will have unexpected event that will disturb the procedure that we have plan that think will work "perfectly" executed during the project.

If we can have more time to develop this program, at first we would develop a GUI interface for the users to use. The reason why we would develop Graphical User Interface is when the phone organizer can have more interaction with the program. The benefit of using GUI interface is can let more user to learn to use the program more conveniently and especially for the peoples who have less educated, they can't read those complicated words but they can know the meaning by seeing the pictures of each options.

Then, the user can have their own contact book which is not support at this stage, for them having a private and unique contact list, this can let them protect their privacy which is a valuable data to the users. The data stored by the program also can encrypted to make sure no one can have unauthorized access to the contact file.

The program can also having data validation to the input data when more time is allowed. For example, the email address of input data must have a "@" symbol so that it can sure the user will not input the wrong data, the input date and phone number cannot have any characters, even the date can check the users is or is not input the wrong data, i.e. 22th September, 2015 05:34pm input as (MMDDHHSS) 22110534 which is no 22th month in reality but the 12-24 hours can't be done because 0534am is also a valid input. Or even don't allow the same person (who have the own phone number, email and name) to input the data twice. The program will prompt a message and suggest the user to delete the old data or update the old data.

That's will let to another function that will provided if there is more time – the alter function of the contact data. Now, the program only provide add a new data and delete the data as this is the requirement from this assignment. But for the user to update or alter their own data input is also a necessary function to make this

program more convenient to the users. For example, when the user have call someone after the input times, the users need not to delete the data and input it from nothing at start, the user can just simply update the last call time and save time and efforts for the user to input the name, phone number, email... again and again. User friendly plays an extremely important parts of a good program.

Also, we would like to create a library that's allow other program to use the contact store inside the book such as a social media application can directly use our program and alter the data that the contact last call time or even can store the Facebook, Twitter, Instagram etc. information inside the program, the users can also share their own contacts just with one clicks to imports and exports others information.

Moreover, the phone organizer will also provide much more customization if more time is allowed, instead of the user only can choose from Family, Friend and Junk, the user can create their only groups such that the record can be even more organized.

Last but not least, a cloud-based apps is a modern trend of applications. When the program is cloud-based, that means the program will become cross platform, no matter the user is using computer of Windows, Macintosh or even using smartphone of iOS , Android or windows phone can use the applications. This is more continent for the users because the phone organizers is mainly use in the mobile phone, the user can use the applications right after they use the phone to have a phone call. Also, a cloud-based program will have another benefit as if we have a server, all of the users data can be backup, and the need not to use the same device to read back the contact, all data will be synchronized across all platform.

When the program become web-based, it is much easier to maintenance as the program can be updated through the web server, it does not require to remind the user to update the program to have bug fix or add a new feature.