**EEE101 C Programming and Software Engineering 1 – ASSESSMENT 6**

**Project B: Bank Information System**

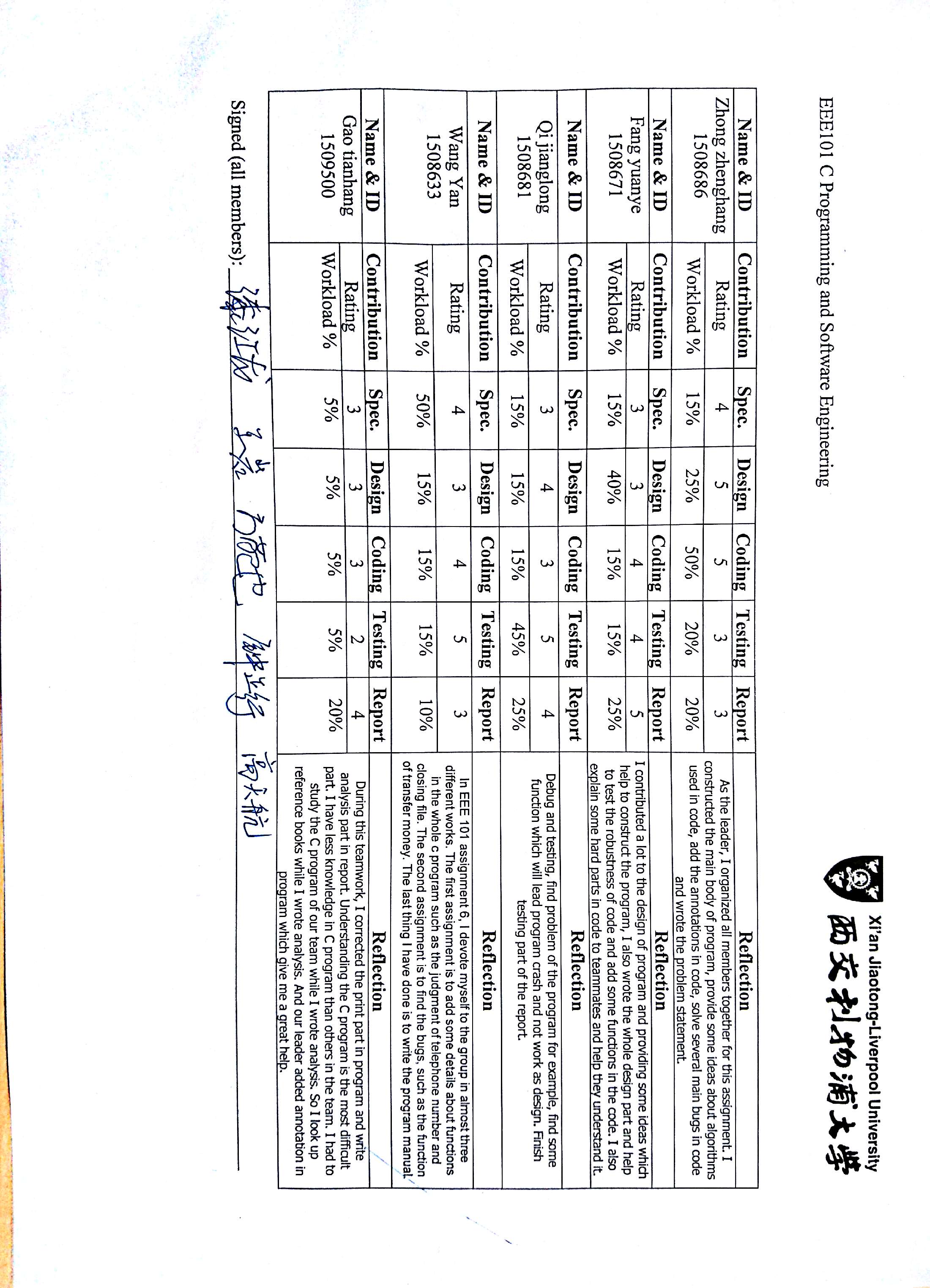
**Group number: group 8**

**Contribution form**

All groups must submit the following contribution form as part of their report for Assignment 6. The form should be:

* The second page of your report (After the title page, you can use more than 1 page)
* Signed by all the group members  
  Any group members who do not contribute to the assignment or have made no effort to communicate with the group can be noted here.
* Breakdown of the roles and responsibilities of each group member during the duration of the project.  
  **Rating** – Students should sincerely rate their contribution on a scale between 1 (low) to 5 (high)  
  **Workload** – Students should determine the percentage of each section they have completed (total for all students should be 100%)

Reflection – should be your individual refection or comments about your work during the group (no more than 70 words per student).



1. **Problem statement.**

Design a bank system program. Customers, bank clerks and managers are able to set up an account and log in with account information. Moreover, they have following functions.

Design a bank system program. Customers, bank clerks and managers are able to set up an account and log in with account information. Moreover, they have following functions.

Firstly, customers are asked to entry their name, address, telephone number and pin, then check if the telephone number is unique (if not unique, customers will be asked to entry again, as in this program telephone number is used to identify different customers, like ID), and the information including balance which could only be deposited by clerks, the state of active or inactive could only be changed by manager (but it is built up active), then the program will produce a random 8-digit card number which is used to log in combing with pin. After logging in successfully, customers could choose six modes and before acting every mode they are asked to enter pin for security, the customer could choose from withdrawal balance, access to customer information (partially, not including the state information and pin), change pin number, transfer money to others, access to activities recordings and log out. All the activities about customer need to be recorded in a special file and could be accessed and printed.

Secondly, bank clerks only need name and password to log in. Then they have the ability to add accounts for existing customers who are identified by their telephone numbers, the ability to delete accounts according to card ID and amend accounts’ information like name and address, also they could make deposits into an account which need to be larger than 0. Moreover, clerks could set up the standing order, which means set the function where customer A transfer to customer B some decided money in decided frequency as long as customer A has enough balance.

Thirdly, managers only need to log in with name and password. Then they could choose from 8 modes, and they have the ability to access to customers’ information including name, address, and telephone number, state, and balance and so on; Managers could also block or unblock customers according to card ID and meanwhile they could access to some big data, like the number of customers, number of accounts and average accounts per customers; and the total balance and average account balance could also be accessed.

Generally speaking, customers, clerks and managers need to be divided into different parts to be designed and work together smoothly. Customers need to remember the system-produced random card ID and 6-digit pin to log in and could perform well when the state is active, which is under control of manager, also one customer could only set up one account with one unique telephone number and more accounts could only be added by clerks, also only the clerks could deposit money to customers and set standing orders between customers. Meanwhile, customers could not change personal information except pin, but clerks could. So customers can not perform well without the help of clerks and managers.

1. **Analysis**

**Customer**

Input:

*Create an account:*

A string to represent the name of account

⊙ Differentiate different customers

⊙ Record the number of customers so that manager can calculate the average data

A string to represent the telephone number of account

⊙ Differentiate different accounts

⊙ Representative an account for different customers might have the same name

⊙ It’s too difficult to use identity card number as the representation of account

A string to represent the address

⊙ Record customers’ information and insure it’s real

⊙ Guarantee the financial safety of bank when customers break their promise

A string to represent the password of account

⊙ Customers set the password by themselves instead of randomly generated an initial password so that customer need not to worry about losing or revealing password before change it

A number of type **long long** to represent the card ID

⊙ Divide money in an account to different cards to make convenience

A string to represent the password of ID card

⊙ Use different password for account and card ID to protect the safety of customers’ property

A number of type **long long** to represent the money in ID card.

⊙ It‘s convenient and clear to use a number of type **long long** to record change of money in an account

*Log in:*

A string to represent the ID card

⊙ Ensure which card that customer want to operate on

⊙ All card are subordinate to the account, this operation will be record in the account

⊙ To simplify the procedure for card ID can already ensure an account

A string to represent the password of account

⊙ Ensure that there the right customer who is the owner of this card

⊙This procedure has nothing to do with money and account so it need not more verification account password

*Withdraw from account:*

A string to represent the password of account

⊙ Ensure people who withdraw is the owner of card

A number type of **long long** to represent the money in card ID

⊙ Let customers know how much money they still have in this card

⊙ Provide the reference so that customer can change the amount of money which they will withdraw

A string to represent the amount of money that you draw

⊙ Customer can input the amount of money that they want to withdraw

A number type of **long long** to represent the amount of money that you draw

⊙ Give customer another chance to ensure the amount of money

A number type of **long long** to represent the money in card ID

⊙ Let customers know the amount of money in this card now

*Check information:*

A string to represent the password of account

⊙ Ensure the identity of people who want to check account information

⊙ Protect the safety of account for customers

Print the information of account (card ID/address/account name /account balance)

⊙ Let customers check the account information and decide change it or not

*Change password:*

A string to represent the password of account

⊙ Ensure the identity of people who want to change the password of account

⊙ Protect the safety of account for customers for this procedure involve the information of card’s password

A string to represent the new password of account

⊙ Customers input the new password of card ID

A string to represent the new password of account

⊙ Give customers a chance to check whether they input right password in the front procedure

Print the time when password changed

⊙ Record the important time point so that customers can check it if others change the password

*Transfer money to another card ID:*

A string to represent the password of account

⊙ Ensure the identity of people who want to check account information

⊙ Protect the safety of account for customers for this procedure involve money in account

A string to represent the card ID which transfer to

⊙ Customers need to input the card ID that they want to transfer to

A string to represent the amount of money to transfer

⊙ Customer need to input the amount of money that they want to transfer to

A number type of **long long** to represent the money in card ID

⊙ Let customers to know how much money they still have in this card ID

A number type of **long long** to represent the money that transfer

⊙ Let customers to check if they transfer right amount of money to another card ID

A number type of **long long** to represent the money that receive

⊙ Show the amount of money that another card ID receive

*Log out:*

An **int** number to quit the program

⊙ Customers log out of system after finish their procedure

**Clerk**

*Create an account:*

A string to represent the name of account

⊙ Clerks input their name as representation of account

⊙ The probability that two different clerks in one bank have the same name is very low so that clerks can just use their name as account

A string to represent the password of account

⊙ Clerks input the password of account

*Log in:*

A string to represent the name of account

⊙ Ensure account that clerks want to operate on

⊙ There are no subordinate relationship in clerks’ account so that log in with name is ok

A string to represent the password of account

⊙ Ensure the identity of clerks

*Add account for a existing customer:*

A string to represent the telephone number of the customer who want to add account.

⊙ Input the telephone number of people who want to create a new account

⊙ Telephone number is the representation of account, so a new telephone can create a new account.

A string to represent the password

⊙ Input the password of customers’ account to complete the creation of new account.

*Delete account for an existing customer:*

A string to represent the card id that want to delete

⊙ Input the card id that customers want to delete

*Amend accounts information:*

A string to represent the card ID that want to amend

⊙ Input the card ID which customers want to amend

⊙ Clerks can relate card ID and account when they know card ID

⊙ But clerks cannot relate card ID an account when they know account for there are many card IDs belong to one account

A string to represent the name that be amended

⊙ Input the new name that customers want to change to

A string to represent the address that be amended

⊙ Input the new address that customers want to change to

**Make deposits into an account**

A string to represent the card ID that want to deposits money in.

⊙ Input the card ID that customers want to deposits money in

⊙ Card IDs belong to the same account are independent of each other ,so input card ID is ok

A string to represent the amount of money that want to deposits in

⊙ Input the amount of money that customers want to deposits in

**Set up standing order**

This function has not been constructed well due to time and ability limitation.

**Log out**

An **int** number to quit the program

⊙ Clerks log out of system after finish their work

**Manager**

**Create an account:**

A string to represent the name of account

⊙ Managers input their name as representation of account

⊙ The probability that two different managers in one bank have the same name is very low so that managers can just use their name as account

A string to represent the password of account

⊙ Managers input the password of their account

**Log in:**

A string to represent the name of account

⊙ Ensure account that managers want to operate on

⊙ There are no subordinate relationship in managers’ account so that log in with name is ok

A string to represent the password of account

⊙ Ensure the identity of managers

*Access customer account information:*

A string to represent the card ID that want to access

⊙ Input the card ID which managers want to access

Print the information of this card ID

⊙ Print the information of this card ID to decide whether block this card or not and contact with card’s owner to inform him/her information such as “pay off the debt as soon as possible”

*Block or unblock accounts:*

A string to represent the card ID that want to access

⊙ Input the card ID which managers want to block/unblock

Print this card ID is active or not

⊙ Ensure the state of this card to do next procedure (block or unblock it)

Block or unblock this card ID

⊙ Block this card ID to protect the profit of bank

⊙ Unblock this card ID for owner’s has already took right reaction

*Number of customers:*

Print the number of customers

⊙ Print the number of customers from system through checking how many name (with different telephone number belong to) in the system

*Number of account:*

Print the number of accounts in bank

⊙ Print the number of account in bank by check how many different telephone numbers in the bank system

*Average account per customers:*

Print the average accounts per customer

⊙ Print the average account per customer through dividing the number of telephone number in bank system by the number of name (with different telephone numbers belong to) in bank system

*Total account balance:*

Print the total balance of bank

⊙ Print the total balance of bank by add the amount of money in every account.

*Average balance:*

Print the average balance of bank

⊙ Print the average balance of bank through dividing the total balance of bank by the number of account.

*Log out:*

An **int** number to quit the program

⊙ Managers log out of system after finish their work

1. **Design**

**3.1 Flowchart**

Create an account for customer:

Input: Name

Address

Tel

Pin

Random card ID

Generate an account

Print it in a file

Log in as customer

Input: Card ID + Pin

Wrong

Withdraw from the account

Input: Pin

Access to account information

Change the pin number

Transfer money

Access to activities record

Create an account for clerk:

Input: Name

Password

Generate an account

Log in as clerk

Input: Name + Password

Wrong

Right

Log out

Make Deposits

Set up standing order

Amend account information

Delete an account

Add accounts

Log out

Welcoming Interface

B

A

N

K

S

Y

S

T

E

M

Record these activities in a file

Record these activities in a file

Create an account for manager

Input: Name

Password

Log in as manager

Wrong

Right

Access customer account informantion

Block or Unblock customer accounts

Number of customers

Number of accounts

Average accounts per consumers

Total balance

Average account balance

Log out

All activities done by the manager should be recorded in a file

**3.2 Design**

This program created four files: file 1 is to collect the data about the customer; file 2 is to collect the data about the clerk; file 3 is to store the data about the manager and file 4 is to record the activities information that people did in this bank system.

Algorithm:

1. --Declare a function called **addCustomer**. This function is to let the user be able to create a new account for customer. This defined function has two arguments in the type of char which points to the address of file 1 in binary mode which records the information of customers and the address of file 2 in text mode which records the data about the activities have been done by the users.
   1. Declare two variables namely account and person both in the type of structure customer. Then use the function of typedef to give the structure customer another name Customer. The **Customer person** stores the information entered by the user and the **Customer account** stores the information of the account in the file 1 one by one to check whether the accounts in the file have the same telephone number or not (in this bank system, our group ask the system to identify the estate of each person by their telephone number). Meanwhile, the **Customer person** also collects the information about the person such as address, name and so on. If the account can be built successfully, eight random numbers representing the ID card of this account will be printed on the screen and customer should remember this ID card to log in next time.
   2. Declare a FILER pointer called **fptr** that stores the address of the file which needed to be opened.
   3. Use the function of **scanf()** to collect the name, address, telephone and pin of a new account a string variable or a variable in the type of **long long**. This system should also judge the return value of **scanf()** to see if the user input a valid variable and simultaneously, the length of the string should also be checked if accord with requirements, and use the function of **fflush(stdin)** to clear the standard input buffer. In addition, to apply the **linked list** to this program, in the defined function our group also added a variable in the type of structure and used it as a pointer to connect the structure in the file.
   4. After storing all the data in the **Customer person,** open the file 1 and write the data in the final space. Open the file with mode “ab+” (users could read and write the information in the binary mode optionally) and use the file pointer namely **fptr** to store the address. Importantly, this program also checks the value of **fptr** to see if the program opens the file successfully, if not, exits the program in case cause more damages to the file. If all goes well, use a function namely fread() to read the a space of memory of the size of **Customer** in the file pointed by **fptr** and store the data in the account in the type of **structure customer**. At last close the file by using the function of **fclose()**. Next open the file 4 used to record the activities done by users and then store the data in it. Return number 0 to end the function.
   5. Use the function **srand(), time() and rand()** to generate the random ID card for the account of customer. Meanwhile, declare a variable **state** in the **structure customer** to let the account be able to be blocked if the state does not equal to 0.
2. --Declare a function namely **logCustomer.** This function also has two pointer inputs of type char which points to the address of file 1 and file 4. This function ensures users to log in with correct information.
   1. To get the time for recording when the user does the activity, this program utilized the function of **time()** and **localtime()**. In addition, to avoid something wrong happening in the program, our group initialized the variable in this defined function.
   2. Declare a variable in the type of long long namely **wMoney** to store the amount of money that the customer wants to withdrawal and another variable namely **tMoney** representing the money that the customer wants to transfer to other person. At the same time, declare some variables in the type of char used to store the information input by the user. Like did in the function **addCustomer**, use the function **scanf()** to store these variables and when the data collected before are needed, open the file and search through the whole file to see if the information matching the account in the file.
   3. Ask the customer to input the card ID and pin to log in his or her account. If the information that they input do not match the data in the file or their accounts have been blocked, tell them there is something wrong. If the information that they input is correct, offer 6 options to them and let them choose.
   4. If the customer wants to withdrawal from their account, ask them to input their pins again and check if it is correct. If it is right, use **wMoney** to store the money that they want to withdrawal and do the operation in the file 1. For the option 2 access to the account information and option 5 access to the activities record, the operations are almost similar except the former is to do the operations in the file 1 and the latter is to do the operations in the file 5. For option 3 change the pin number of the account, the program should ask the user to input the new pin twice and check them. Then use **fopen()** in the more **“rb+”** and the function **strcpy** to rewrite the data in the file1. For option 4 about transferring the money in the account to someone else, it is perhaps the condition that combined the case 1 and case 3. The program should also tell how much money left in the account and permits the money transferred is in the range of money that the customer could transfer. For the last option, the program **return 0** to let the interface back to the initial one.

Note that before each operation except the last one, the customer should input the pin again to do the latter operations and every activity should be written in the file 4 as record.

1. --Almost the same as the defined function **addCustomer**, this program also declared two functions namely **addClerk** and **addManager** to permit the function about creating new account for the clerk and manager. The differences are that the operations in the defined function **addClerk** are in the file 2 and the operations in the defined function **addManager** are in the file 2. In addition, the information that people need to offer to get an account of the clerk or manager is just the name and the password.
2. --Declare a function namely **logClerk**. This function has three pointer arguments in the type of char to connect the data in the file 1, file 2 and file 4 to application. To use **linked list** in this function, the programmer declared three pointers in the type of **structure customer**: **\*head**, is the head address of the structure in the file; **\*next**, is the address used to point at the head address of the next structure; **\*prev**, is the address of the last structure compared with the current structure. Some temporary variables were also declared in this function to store the data and offer convenience to do the operations without the danger of losing or mixing up data in the file.
   1. Firstly use the function **scanf()** to collect the information that the person input about the name and the password. The same as the function **logCustomer**, declare a FILE pointer namely **fptr** and use the function **fopen()** in the mode **“rb”** and **fclose()** to check if the information matches the data in the file or not nad tell the user if there is something wrong.
   2. Declare another FILE pointer namely **fp** with the help of the function **fopen()** in **“rb”** mode and **fclose()** to ensure the file 1 is not empty to avoid the program crush in some cases.
   3. Use the function of **malloc()** to give some memory to the **prev** and then create a **linked list** to store all the structure information.
   4. If the information input by the person is right, offer him or her 6 options required in the task sheet. For option 1, adding a new account for a person. Store the information that the user input in the temporary variable and check whether there has existed a customer account by the identification of the telephone number. With the help of the function **strcpy()** and the **linked list**, copy the information in the previous account that the person have had except the money and the state (whether been blocked or not). Then write the data in file 1. For option 2, delete the account, our group utilized the **linked list** to let the structure which store the information “disappear” and use the function **fopen()** in the **“wb+”** mode to overwrite the data in the file 1. For option 3, to let the clerk be able to amend the information of the customer about name and address, our group decided to cope the input by the user to a temporary variable with the function **strcpy()** and then write the information in the file 1. For option 4, after several check about whether the input by the clerk matches the information in the file 1, our group designed the of transferring money from one account to another by using the knowledge get from assignment 5 and also the help of **linked list**. Be similar to the last defined functions, our group let the user log out by returning 0 to this function.

With the limitation of the time and ability, our group **did not achieve the function about standing order of option 5** but we have some ideas. To create a new file and write all the operation of the standing order into this file and every time the program opens, the program need to access to this file and call the system time as well as change the balance of related accounts.

* 1. Note that after each operation except logging out, the information about these activities should be written in the file 4 with the function **fopen()** in the mode **“a+”** to allow people read the text.

1. --Declare a function namely **logManager**. This function is almost the same as the defined function **logClerk**. It has three pointer arguments in the type of char to connect the data in the file 1, file 3 and file 4 to application. In addition, it also uses linked list to achieve some functions required in the task sheet.
   1. Firstly use the function **scanf()** to collect the information that the person input about the name and the password. The same as the function **logCustomer**, declare a FILE pointer namely **fptr** and use the function **fopen()** in the mode **“rb”** and **fclose()** to check if the information matches the data in the file or not nad tell the user if there is something wrong.
   2. Declare another FILE pointer namely **fp** with the help of the function **fopen()** in **“rb”** mode and **fclose()** to ensure the file 1 is not empty to avoid the program crush in some cases.
   3. Use the function of **malloc()** to give some memory to the **prev** and then create a **linked list** to store all the structure information.
   4. If the manager log in successfully, then the system offers 8 options to him or her. For option 1, access the customer information, is using the card ID of the customer to retrieve data in the file 1 and then read the information store in it with the help of file function. As for option 2, is just change the value of the variable **state** in the file 1 with **fopen()** in the mode **“wb+”**. For option 3, get to know the number of customers totally in the bank, our group achieve the goal by reading the linked list with two structure pointer. If there is the same telephone number in the list, the variable namely **numCustomer** represented the total amount of the customers in the bank will not plus one. In contrast, plus one if there does not exist the same telephone number and stop when the former structure pointer get the address of **NULL**. For option 4, read the total amount of the accounts in the bank, our group did this by using the **while()** loop to read the structure in the linked list until get the end of the **linked list, NULL**. And option 5, get the value of the average amounts per accounts, is just combining the codes of case 3 and case 4 together. Option 6 to get the total money stored in the bank， the code is almost the same as the case 4 unless need to plus the balance in each account after every execution of the **while()** loop. Option 7, to get to know the average account balance in the bank, is just combining the operations in case 4 and case 6. To log out and return back to the initial interface, still return 0 to this function.
   5. After all the operations, free the linked list using function **free()** (free the linked list from head to end).
2. –In addition, some codes used to avoid invalid input were also added in this system. After the invalid input each time, our group used **fflish(stdin)** to clear the stander input buffer.
3. **Implementation**

See the C code in file EE101\_Group8\_bank system.c with comments.

1. **Testing**

During the time we code, we had actually encountered with several problems and find some deficiency aspects with some of them successfully solved others not solves well due to the limitation of time. I will list them below:

**1 main interface**

The first problem we have solved is about the user interface. There are 7 choice were to user. However, when the input digit out of 1 to 7 the program will crashes. This problem solved by instead the function “scanf("%d",&option)!=1 && (option<1 || option>7)” by “scanf("%d",&option)!=1 || (option<1 || option>7)”. Make the logic correct. When the digit out of range the program will remind user.

**2 create part**

2.1 The first part the program, create account part. If the user wants to create customer account they need to input their name phone number address and pin, and in the end they will get their card id which generated by function “srand()” followed by function “fread()” and “strcmp()” to make sure their card id is unique.

2.2 During the create part. Every data that input have their own limit of the number of character. Function “strlen()” will make sure that the length will not out of range. If out of range the program will remind them and let them input again.

2.3 The telephone number of customer and name of clerk and manager are the variable that ensure their will not exist repeated account, and they all work by use function “strcmp()” and “fseek()”.

**3 customer part**

3.1 Log as customer require the card ID and pin. There are 6 function given to customer. As well as the beginning user interface, when you input a number out of range the program will remind you.

3.2 Every function of the customer require the pin to access. This step makes the process slow, but it’s the way that we decide protect account.

3.3 Withdrawal and transfer function related to balance. So the program will show their balance at that time, and when the money that customer wants to operate lager than the balance the program will tell them. After operation the program will show customer the balance. Transfer function need the user to input card ID of the one they want to transfer money. The program will notice the user when the card ID form is incorrect or ID not exist.

3.4 The second problem we found is that the balance and pin will not change after user withdrawn money and changed the pin. Although it changed and saved but the information shows the original data. This problem solved by refresh structure “account” after operation. Right now, user can see the newest data just after what they had done.

**4 clerk part**

4.1 The third problem we found was that account information would not change after the user log in as clerk and changed it. To solve this problem we put the function “strcpy()” into every case instead of we put it outside of the loop “while()”. So the newest data will show to user just after what they changed.

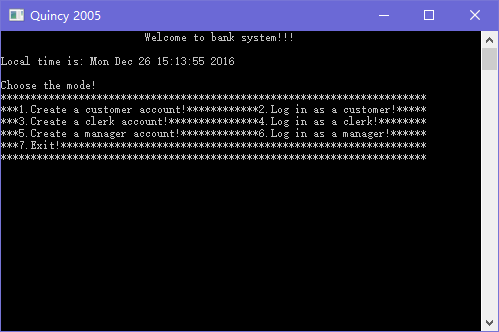
4.2 The fourth problem we found was that user cannot use the amend account information after the user changed the account information and quit. The problem result to variable “option2” had not been initialize after quit which makes the “option2” equal to 3. Makes the program will quit immediately after choose amend account information function.

**5 manager part**

5.1 The fifth problem was that the program will crash after some functions of manager. This is caused by at beginning try to use “while” function to judge structure pointer, and it solved by use “if” function to judge structure pointer.

The C program was tested by carrying out a set of experiments; and the C program output was verified successfully. For instance,

**6**  **The interface of main menu**



6.2 When I input number out of range, program would say “Please enter again and make sure it is valid input”.

6.3 When I create a customer account with telephone number already exists, program would say “This telephone number has been used before. Please try another one!!!” and return to previous user interface.

6.4 When I create a clerk or manager account with name already exists, program would say “The name has already been used!! Try another one!” and return to previous user interface.

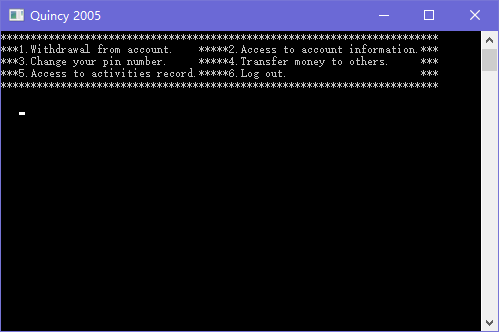
6.5 When I log in as a customer with a card ID that not have 8 digit, program would say “Please try again and make sure the card number are equal to 8-digit” and let input again.

6.6 When I log in as a customer with a card ID that does not exist, program would say “No such an account exist!” back to the previous user interface again.

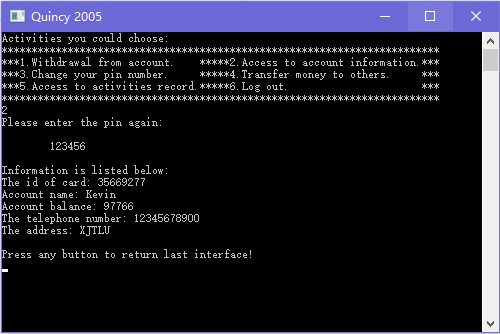
6.7 When I log in as a customer with a pin that not have 8 digit, program would say “Please try again and make sure the card number are equal to 6-digit” and let input again.

6.8 When I log in as a clerk with the intent to change customer information. The number I input out of range the program would say “Please try again and make sure your input is 1, 2 or 3” and let user to input another digit.

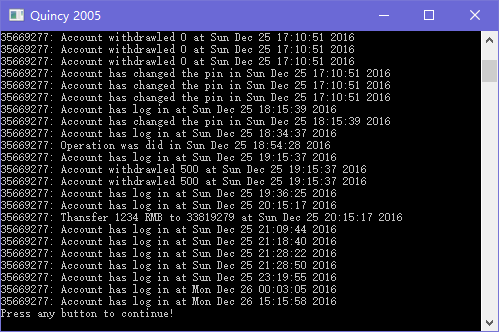
6.9 The interface of customer menu is like below:



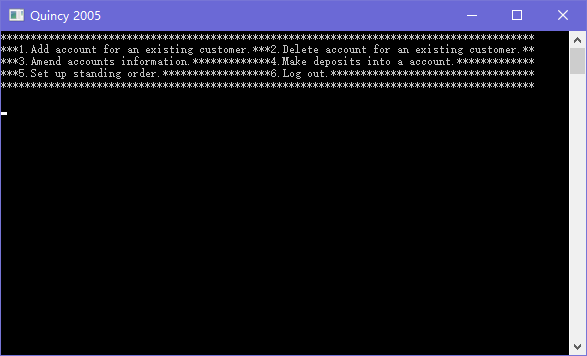
6.10 The interface of customer information is like below:



6.11 The interface of customer activities record is like below:



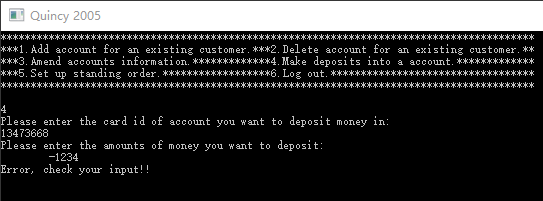
6.12 The interface of clerk menu is like below:



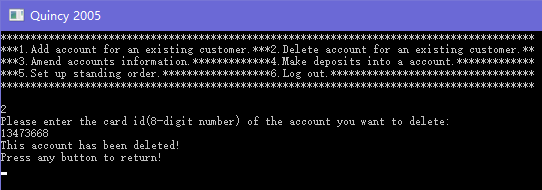
6.13 The interface of amend account information menu is like below:



6.14 When the user input negative number in deposits function the program will show some information like below

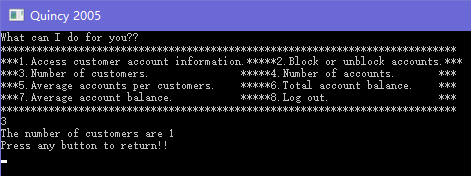


6.15 The delete account function are show as below:

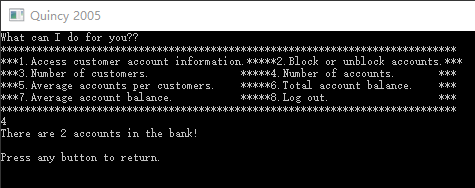


6.16 Set up standing order function have not finished yet.

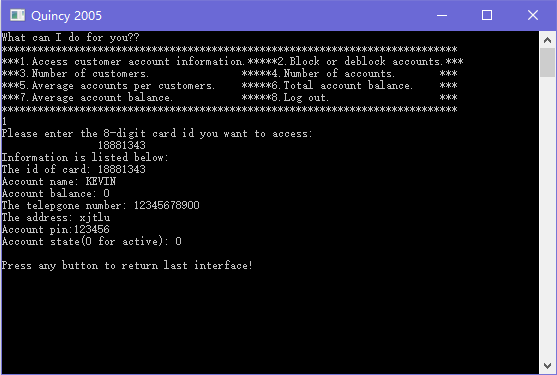
6.17 The count customer function will show total customer of the bank:



Function number of accounts will show all the account in the bank:

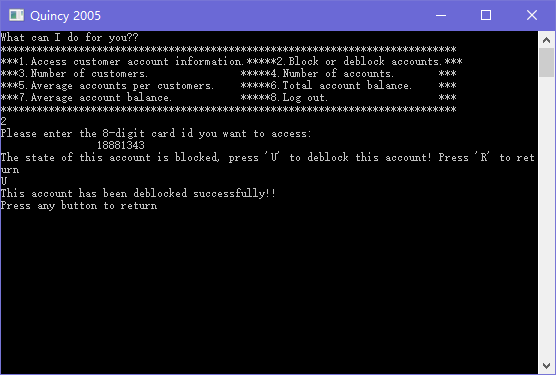


6.18 The interface of access customer account information menu is like below:



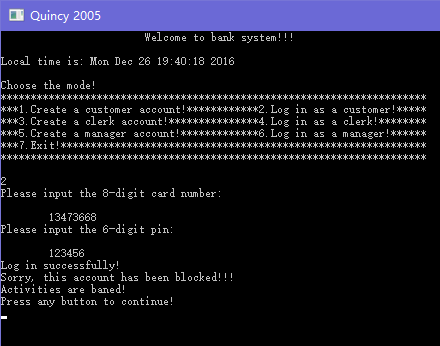
The account state 0 means the account is active, 1 means blocked.

6.19 Block and unblock accounts function.

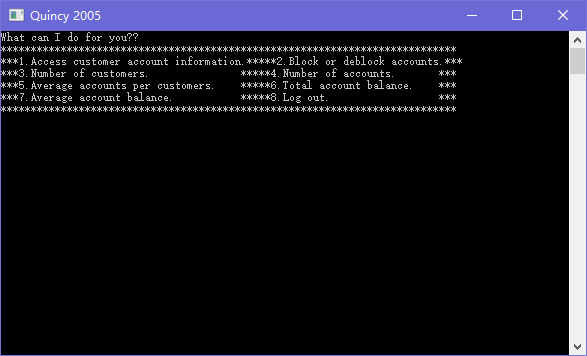


The function Block and unblock accounts interface like upon. And it works as press “B” button to block and press “R” to unblock account. Press “R” button can return to manager interface.

6.20 When the account had been block, will give the user information like below:



6.21 The interface of manager menu is like below:



6.22 The average accounts per customers function, total account balance function and average account balance function are show below:

