Banking System Manual

Contents used:

Libraries.
Global Variables.
File Pointers.
Input Variables.
Arrays and Strings.
Functions.

Functions used:

countLines(): Counts the number of lines in the "accounts.txt" file
Load(): Loads account data from the "accounts.txt" file into an array of structures to use them in other functions.
Login(): Authenticates the user by checking his username and password in "users.txt" file

```
Hello User !
--------
(1)Login
(2)Quit
--------
1
----Login In----
Enter Username: aaa
Enter Password: 123a


----Successful Login!----
```

isValidAccNum(): Checks if an account number is valid according to the requirement of being account number.
convertMonth(): Converts a month number to its corresponding string representation
Quit(): Exits the program
IsValidName(): Checks if a name contains only alphabetic characters
capitalize(): Capitalizes the first letter of a name

Advanced_Search(): Searches for an account by name

```
Enter Name: michael

Account Number: 9700000000
Name: Michael Jones
E-mail: m.jones@gmail.com
Balance:1000 $
Mobile:01009700000
Date Opened: December 2007



Account Number: 9700000003
Name: Michael Robert
E-mail: michael@yahoo.com
Balance:300 $
Mobile:01009700003
Date Opened: November 2008
```

Add(): The function prompts the user to enter account details, such as number, name, email, balance, and phone to create new account.

```
----Choose Option----
(1) ADD
(2) DELETE
(3) MODIFY
(4) SEARCH
(5) ADVANCED SEARCH
(6) WITHDRAW
(7) DEPOSIT
(8) TRANSFER
(9) REPORT
(10) PRINT
(11) QUIT
----------------
==>1
Please enter the account number to be added:
9700000020
Please enter the account user's name:
ahmedkamel
Please enter the account user's email:
a23@gmail.com
Please enter the account user's intial starting balance:
20000
Please enter the account user's phone number:
01140485888

Do You Want To Save Changes?
(1)Save
(2)Discard
```

If you print after adding you will found the client you have just added:

```
Account Number: 9700000003
Name: Michael Robert
Email: michael@yahoo.com
Balance: 300
Phone: 01009700003
Date Opened: 11-2008


Account Number: 9700000007
Name: Philipe Brian
Email: p.brian@outlook.com
Balance: 460
Phone: 01009700007
Date Opened: 2-2020


Account Number: 9700000004
Name: Roberto Thomas
Email: rob.thomas@gmail.com
Balance: 400.5
Phone: 01009700004
Date Opened: 11-2015


Account Number: 9700000002
Name: Timothy Korman
Email: t.korman@gmail.com
Balance: 200
Phone: 01009700002
Date Opened: 12-2015


Account Number: 9700000020
Name: ahmedkamel
Email: a23@gmail.com
Balance: 20000
Phone: 01140485888
Date Opened: 12-2023
```

Delete(): It checks for the existence of the account and ensures its balance is zero before deletion.

Before deletion:

```
9700000000,Michael Jones,m.jones@gmail.com,1000,01009700000,12-2007
9700000001,John Roberto,j.roberto@outlook.com,100,01009700001,12-2008
9700000002,Timothy Korman,t.korman@gmail.com,200,01009700002,12-2015
9700000003,Michael Robert,michael@yahoo.com,300,01009700003,11-2008
9700000004,Roberto Thomas,rob.thomas@gmail.com,400.5,01009700004,11-2015
9700000005,David Roberts,david123@gmail.com,400.5,01009700005,10-2015
9700000006,Daniel Graves,dgrave@outlook.com,450,01009700006,1-2020
9700000007,Philipe Brian,p.brian@outlook.com,460,01009700007,2-2020
9700000008,Adam Mark,ad.mark@gmail.com,350,0100970008,10-2015
9700000009,James Adams,j.adams@gmail.com,0,01009700009,5-2017
```

Deletion:

```
Please enter the account number to be deleted:
9700000007

Do You Want To Save Changes?
(1)Save
(2)Discard
1
Account deleted successfully.
```

After deletion:

```
9700000000,Michael Jones,m.jones@gmail.com,1000,01009700000,12-2007
9700000001,John Roberto,j.roberto@outlook.com,100,01009700001,12-2008
9700000002,Timothy Korman,t.korman@gmail.com,200,01009700002,12-2015
9700000003,Michael Robert,michael@yahoo.com,300,01009700003,11-2008
9700000004,Roberto Thomas,rob.thomas@gmail.com,400.5,01009700004,11-2015
9700000005,David Roberts,david123@gmail.com,400.5,01009700005,10-2015
9700000006,Daniel Graves,dgrave@outlook.com,450,01009700006,1-2020
9700000007,Philipe Brian,p.brian@outlook.com,460,01009700007,2-2020
9700000008,Adam Mark,ad.mark@gmail.com,350,0100970008,10-2015
```

Modify(): Modifies information in an existing user account.

Search(): Searches for an account by account number

```
Enter Account Number To Search For: 9700000009
Account Number: 9700000009
Name: James Adams
E-mail: j.adams@gmail.com
Balance: 250
Mobile: 01009700009
Date Opened May 2017
```

Save(): Saves account data to the "accounts.txt" file

Deposit(): Handles the deposit operation for an account

Before deposit:

```
Account Number: 9700000003
Name: Michael Robert
Email: michael@yahoo.com
Balance: 300
Phone: 01009700003
Date Opened: 11-2008


Account Number: 9700000007
Name: Philipe Brian
Email: p.brian@outlook.com
Balance: 460
Phone: 01009700007
Date Opened: 2-2020


Account Number: 9700000004
Name: Roberto Thomas
Email: rob.thomas@gmail.com
Balance: 400.5
Phone: 01009700004
Date Opened: 11-2015
```

Deposit:

```
----Choose Option----
(1) ADD
(2) DELETE
(3) MODIFY
(4) SEARCH
(5) ADVANCED SEARCH
(6) WITHDRAW
(7) DEPOSIT
(8) TRANSFER
(9) REPORT
(10) PRINT
(11) QUIT
-----------------
==>7
Enter Account Number: 9700000007

Enter Amount To Deposit:
2000

Do You Want To Save Changes?
(1)Save
(2)Discard
1

Old Balance Is: $460.00
New Balance Is: $2460.00
```

After deposit:

```
Account Number: 9700000003
Name: Michael Robert
Email: michael@yahoo.com
Balance: 300
Phone: 01009700003
Date Opened: 11-2008

Account Number: 9700000007
Name: Philipe Brian
Email: p.brian@outlook.com
Balance: 2460.00
Phone: 01009700007
Date Opened: 2-2020

Account Number: 9700000004
Name: Roberto Thomas
Email: rob.thomas@gmail.com
Balance: 400.5
Phone: 01009700004
Date Opened: 11-2015
```

Withdraw(): Handles the withdrawal operation for an account

```
Enter Account Number: 9700000009

Enter Amount To Withdraw:
200

Do You Want To Save Changes?
(1)Save
(2)Discard
1

Intial Balance is: 250.00
Current Balance is: 50.00
```

Report(): gives the user detailed report about last 5 transaction.

Report on the previous withdraw:

```
----Choose Option----
(1) ADD
(2) DELETE
(3) MODIFY
(4) SEARCH
(5) ADVANCED SEARCH
(6) WITHDRAW
(7) DEPOSIT
(8) TRANSFER
(9) REPORT
(10) PRINT
(11) QUIT
----------------
==>9
Please Enter Account Number To Generate Report For:
9700000009

The previous transactions are:
200.00 is withdrawn from the account and current balance is 50.00
```

Transfer(): Handles the transfer of funds between two accounts

Before transfer:

```
Account Number: 9700000000
Name: Michael Jones
Email: m.jones@gmail.com
Balance: 900.00
Phone: 01009700000
Date Opened: 12-2007


Account Number: 9700000003
Name: Michael Robert
Email: michael@yahoo.com
Balance: 400.00
Phone: 01009700003
Date Opened: 11-2008
```

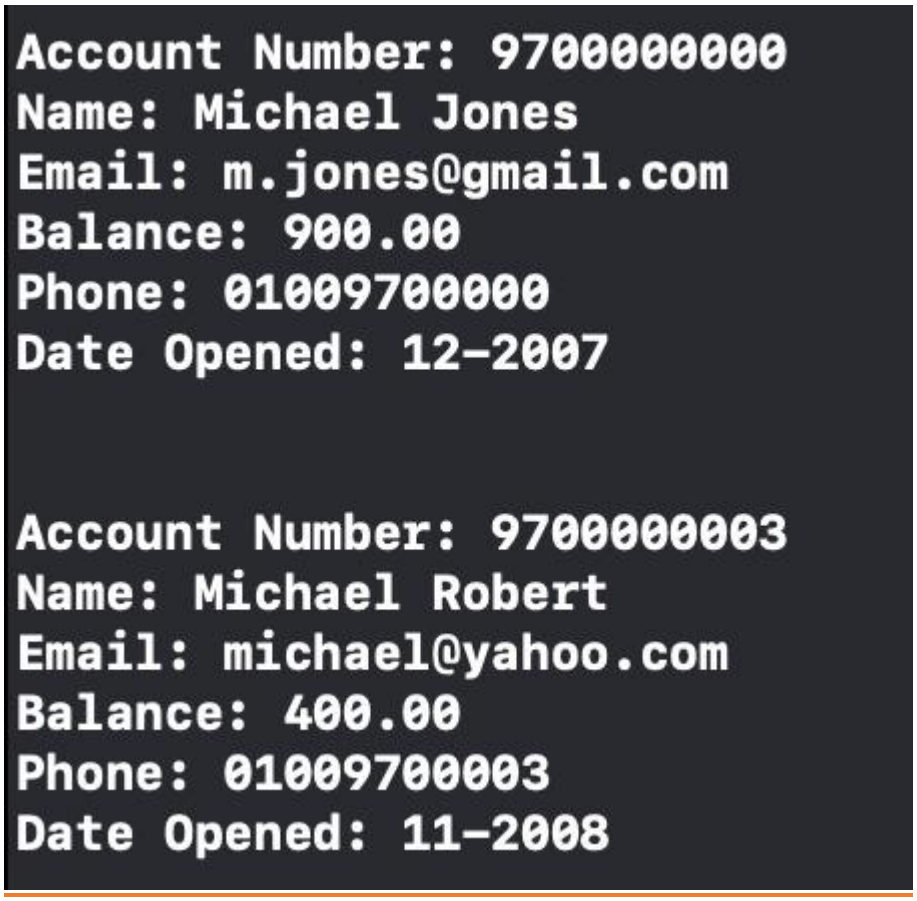
Transfer:

```
----Choose Option----
(1) ADD
(2) DELETE
(3) MODIFY
(4) SEARCH
(5) ADVANCED SEARCH
(6) WITHDRAW
(7) DEPOSIT
(8) TRANSFER
(9) REPORT
(10) PRINT
(11) QUIT
----------------
==>8
Enter Donor Account Number: 9700000000
Enter Receiver Account Number: 9700000003
Enter Transfer Amount: 100

Do You Want To Save Changes?
(1)Save
(2)Discard
1

Donor Intial Balance: 1000
Receiver Intial Balance: 300
Donor Current Balance: 900.00
Receiver Current Balance: 400.00
```

After transfer:



Print(): Displays a menu to choose the sorting criteria for displaying account
SortByName(): Sorts accounts by name.
SortByBalance(): Sorts accounts by balance
SortByDate(): Sorts accounts by date opened

It can print according to three sorting method:

Sorting by date:

```
----Accounts Sorted By Date:----

Account Number: 9700000007
Name: Philipe Brian
Email: p.brian@outlook.com
Balance: 2460.00
Phone: 01009700007
Date Opened: 2-2020


Account Number: 9700000006
Name: Daniel Graves
Email: dgrave@outlook.com
Balance: 450
Phone: 01009700006
Date Opened: 1-2020


Account Number: 9700000009
Name: James Adams
Email: j.adams@gmail.com
Balance: 250
Phone: 01009700009
Date Opened: 5-2017

Account Number: 9700000002
Name: Timothy Korman
Email: t.korman@gmail.com
Balance: 100.00
Phone: 01009700002
Date Opened: 12-2015


Account Number: 9700000004
Name: Roberto Thomas
Email: rob.thomas@gmail.com
Balance: 400.5
Phone: 01009700004
Date Opened: 11-2015
```

Sorting by balance

```
Email: rob.thomas@gmail.com
Balance: 400.5
Phone: 01009700004
Date Opened: 11-2015


Account Number: 9700000005
Name: David Roberts
Email: david123@gmail.com
Balance: 400.5
Phone: 01009700005
Date Opened: 10-2015


Account Number: 9700000008
Name: Adam Mark
Email: ad.mark@gmail.com
Balance: 350
Phone: 0100970008
Date Opened: 10-2015


Account Number: 9700000003
Name: Michael Robert
Email: michael@yahoo.com
Balance: 300
Phone: 01009700003
Date Opened: 11-2008


Account Number: 9700000009
Name: James Adams
Email: j.adams@gmail.com
Balance: 250
Phone: 01009700009
Date Opened: 5-2017
```

Sorting by names:

```
----Accounts Sorted By Name:----

Account Number: 9700000008
Name: Adam Mark
Email: ad.mark@gmail.com
Balance: 350
Phone: 0100970008
Date Opened: 10-2015


Account Number: 9700000020
Name: Ahmedkamel
Email: a23@gmail.com
Balance: 20000
Phone: 01140485888
Date Opened: 12-2023

Account Number: 9700000006
Name: Daniel Graves
Email: dgrave@outlook.com
Balance: 450
Phone: 01009700006
Date Opened: 1-2020


Account Number: 9700000005
Name: David Roberts
Email: david123@gmail.com
Balance: 400.5
Phone: 01009700005
Date Opened: 10-2015
```

Menu(): Displays the menu of options for user interactions

```
----Choose Option----
(1) ADD
(2) DELETE
(3) MODIFY
(4) SEARCH
(5) ADVANCED SEARCH
(6) WITHDRAW
(7) DEPOSIT
(8) TRANSFER
(9) REPORT
(10) PRINT
(11) QUIT
----------------
==>
```

How to Use:

Compile the code using a C compiler (e.g., GCC)

Run the compiled executable

Choose option 1 to log in or option 2 to quit If logging in, enter a valid username and password

```
Hello User !
--------
(1)Login
(2)Quit
--------
1
----Login In----
Enter Username: ziad.ali
Enter Password: 123abc


----Successful Login!----
```

Once logged in, the main menu will be displayed Choose from various options to perform operations such as adding, deleting, modifying accounts, and more.
Follow on-screen instructions for each option

```
----Choose Option----
(1) ADD
(2) DELETE
(3) MODIFY
(4) SEARCH
(5) ADVANCED SEARCH
(6) WITHDRAW
(7) DEPOSIT
(8) TRANSFER
(9) REPORT
(10) PRINT
(11) QUIT
-----------------
==>
```

Notes

The code uses two text files, "accounts.txt" for storing account details and "users.txt" for storing user credentials and can create files for new user and report files for transactions of user.
Input validation is performed for various fields to ensure data integrity.
There are recursive calls in case of invalid inputs, providing the user with options to try again or quit.
The functions call other functions like **Quit** and **Menu** for program flow control.

Search & Sort Algorithms Used

In the bank management system computer program, there were different algorithms used for searching and sorting user accounts data. In the following section it is provided a description of how these algorithms were used to properly execute program functions.

1. Linear Search

Linear search, also known as sequential search, is a method for finding a target value within a list or array. It involves checking each element of the list one by one until a match is found or the entire list has been traversed.
In Query (Search) function, linear search was used to search for a user account number in an array of structures. Each structure contains the user's account number that is required to search for. In the following section it is provided a step-by-step explanation of how linear search works in the function.
1-Start from the beginning: The search begins by setting an integer variable (flag) by zero to indicate that no account number has been found so far. Then examining the first account number in the first structure.
2-Compare with the target account number: Check if the current number is equal to the target number searched for.
3-Match found: If the current element is equal to the target, the search is complete, and the index or position of the element is known from loop index (i).
4-No match: If the current element is not equal to the target, move to the next structure in the array.
5-Repeat: Steps 2-4 are repeated until a match is found or the end of the array is reached.
6-End of Array: If the end of the array is reached without finding a match, the search is done but no account name is found, and the flag variable will still contain a value of zero.

2. Bubble Sort

NOTE: In the program, bubble sort was used in several functions (SortByName, SortByBalance, and SortByDate). But the same sort algorithm was used regardless of the function.
Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted. It is provided in the following section a step-by-step explanation of how bubble sort was used in the program to sort accounts' structures based on several factors.

1-Start at the Beginning: Begin with the first structure value in the array.
2-Compare Adjacent Elements: Compare the current structure value with the next value.

3-Swap if Necessary: If the current value is greater than the following value, swap them.
4-Move to the Next Pair: Move to the next pair of structures (i.e., advance to the next index).
5-Repeat Until the End: Repeat steps 2-4 until you reach the end of the list. After the first pass, the smallest element is guaranteed to be in its final position at the end of the list.
6-Repeat for the Unsorted Part: Repeat steps 1-5 for the remaining unsorted elements. After each pass, the next largest element will be in its correct position.
7-Continue Until Sorted: Continue these steps until the entire list is sorted.

Notes:

- The number of passes in the worst case will be less than the number of elements by 1.
- In every pass the number of elements iterated on will be the total number of elements subtracted from the number of passes. That is because in every pass it is sure that the last item will be in the correct position.