

Assignment: Process Management and Distributed Computing

Due Date: 11:59pm Friday 21st October 2016

Weighting: 30%

Group or Individual: Group of Two (preferred) or Individual

1 Assignment Description

Task 1:

You only need to attempt this task if you are aiming to achieve a mark up to 64% for this assignment.

A virtual banking provider has commissioned you to develop a client/server online banking system. This will allow registered clients access to their bank accounts online. The banking provider wants a simple interface similar to a console based Automated Teller Machine (ATM).

The following account types will be supported by the ATM.

1. Savings Account
2. Loan Account
3. Credit Card Account

Note – A client may only have a maximum of one (1) of each account type (i.e. a client can only have one Savings Account, one Credit Card Account and/or one Loan Account). These are the only account types supported by the ATM. Some clients may only have one account type (e.g. Savings Account) while other clients may have two (2) different account types (e.g. Loan Account and Credit Card Account) or a client may have all three (3) different account types (Savings Account, Loan Account, and Credit Card Account). If a client has only a single account it may be any of the three possible account types. Similarly if a

client has only two accounts these accounts may be a combination of any of the three possible account types.

You have been provided with four (4) text files. The first file is named “***Authentication.txt***” and this file contains every registered client with their user name, PIN and client number. The user name, PIN and client number is unique for each client. The virtual banking provider has insisted that under no circumstances is this file to be modified in any way. (See Figure 1).

Username	PIN	ClientNo
numtheory	613548	28510631
graphpath	2454	35498215
arclamp	18755	64875412
tenthprob	624789	65451124
cipherman	123151	87154654
nottrue	1975	25418846
fluidflow	6287	35987154
elasticity	1664	97654152
symmetry	92455	15446544
butterfly	6451	34457985

Figure 1 – Authentication.txt

The second file is named “***Accounts.txt***” and this contains a list of all the accounts in the ATM system. Each entry in this file consists of an Account Number, the opening balance for the account and the closing balance for the account. The opening balance is the initial balance when the account was first created. The opening balance never changes and is kept for historical purposes only. The only value that needs to be maintained in “***Accounts.txt***” is the *Closing Balance* for the account. No other information is to be stored in this file. The Account Number order is **not** to be modified. See Figure 2.

AccountNo	OpeningBal	ClosingBal
11012342	7815.16	7815.16
12013464	-256.78	-256.78
13014586	-165.22	-165.22
11012375	9825.23	9825.23
12013500	-10022.15	-10022.15
13014625	-3598.25	-3598.25
11145442	982.05	982.05
12158664	-988.21	-988.21
11174581	12.20	12.20
12190452	-100.25	-100.25

Figure 2 – A representative sample of data located in Accounts.txt file

The third file is “*Client_Details.txt*” and this contains each registered client in the ATM system. This file contains the first name, last name, Client Number and the accounts owned by each client. The accounts each client owns is in a

Firstname	Lastname	ClientNo	Accounts
Carl	Gauss	28510631	11012342, 12013464, 13014586
Leo	Euler	35498215	11012375, 12013500, 13014625
Hertha	Ayrton	64875412	11145442, 12158664
Julia	Robinson	65451124	11174581, 12190452, 13206323
Alan	Turing	87154654	13446565, 14668980, 15891395
George	Boole	25418846	14532364, 15853488, 17174612
Danny	Bernoulli	35987154	14532419
Sophie	Germain	97654152	14544365, 17188795
Emmy	Noether	15446544	17053500, 18474625
Mary	Cartwright	34457985	15754794, 18619302

comma separated list. The file is not to be modified. See Figure 3.

Figure 3 – Client_Details.txt file.

Notice that each client may have as many as three (3) different accounts or as few as one account. The account type may be determined by the Account number. Each account number is unique and not shared between clients.

- Savings Account (Account Number is a multiple of 11)
- Loan Account (Account Number is multiple of 12)
- Credit Card Account (Account number is multiple of 13)

Note: The ClientNo in “Client_Details.txt” links the ClientNo stored in “Authentication.txt”.

The fourth file provided is “*Transactions.txt*”. To enable a full transaction history to be provided to the client for each account type owned by the client all transactions that are completed successfully must be maintained *between server instances*. A single server instance is defined as period from when the server is started until the server is terminated by an interrupt signal. If the server is restarted then this begins a new subsequent server instance. Your implementation must be able to persist data between these server instances. This means that if the server is shut down and then restarted there must be no loss of transactional data.

“*Transactions.txt*” must be maintained to store a historical record of every transaction. Each transaction record must consist of:

- From Account
- To Account
- Transaction Type (Deposit/Withdrawal/Transfer)

- Amount

No other information is to be stored in this file. The transactions are to be sorted in **ascending *From Account Number* order** (Figure 4 shows an example).

FromAccount	ToAccount	TranType	Amount
14532364	14532364	2	-15.54
14532364	17174612	4	-89.24
15754794	15754794	2	-155.26
15754794	14544365	4	-98.65
18619302	18619302	3	652.58

Figure 4 – Transactions.txt file – entries sorted in ascending FromAccount order.

Note the TranType column is a record of the transaction type (Deposit, Withdrawal or Transfer).

- Deposit – TranType = 2
- Withdrawal – TranType = 3
- Transfer – TranType = 4

For Deposits and Withdrawals the *From Account* is always the same as the *To Account* in the transaction record. For **Transfers** the *From Account* is the account where the money is transferred *from* and the *To Account* is the destination account (the account where the money is transferred *to*).

Discussion of supplied text files is discussed in more detail later in this document.

The client and server must be implemented in the C programming language using BSD sockets on the Linux operating system. This is the same environment used during the weekly practicals. The programs (clients and server) are to run in a terminal reading input from the keyboard and writing output to the screen.

You will have acquired all the necessary knowledge and skills to complete the assignment by attending all the lectures and practicals, and completing the associated reading from the text book.

Server

All four (4) text files (*Accounts.txt*, *Authentication.txt*, *Client_Details.txt* and *Transactions.txt*) are to be located in the same directory as the server

executable. The server will take only one command line parameter that indicates which port the server is to listen on. If no port number is supplied the default port of 12345 is to be used by the server. The following command will run the server program on port 12345.

```
./Server 12345 (./Server [port number])
```

The server is responsible for ensuring only registered clients of the banking provider can access the online banking system. The file *Authentication.txt* contains the user names and the personal identification numbers (PINs) of all registered clients that may access the ATM system. A registered client may access only their accounts in the ATM system by using a combination of their user name and PIN.

The server should not be able to take any input from the terminal once it is running. The only way that the server is to exit is upon receiving a SIGINT from the operating system (an interrupt signal). SIGINT is a type of signal found on POSIX systems and in the Linux environment. This indicates *ctrl+c* has been pressed at the server terminal. A signal handler must be implemented to ensure the server exits cleanly. This means the server needs to deal elegantly with any threads that have been created as well as any open sockets, dynamically allocated memory, open files and successfully completed transactions stored in memory. When the server receives a SIGINT it is to immediately commence the shutdown procedure even if there are currently connected clients.

It is a requirement that all transactions that have been successfully completed are stored and sorted in ascending account number order in the file “*Transactions.txt*”. In addition if the *closing balance* of an account has changed during the server instance this information must be updated in “*Accounts.txt*”. Your implementation should consider how to minimise the overhead incurred by file I/O. (This is discussed further at the end of Task 1 – Implicit Functionality).

In the 9th edition of the textbook, Section 4.6.2 Signal Handling on page 183 has an introduction to signal handlers and Section 18.9.1 Synchronisation and Signals on page 818 has a detailed description of signals on Linux.

[Beej's Guide to Networking](https://beej.us/guide/bgipc/output/html/multipage/signals.html) has a good discussion with numerous links on signals which may be found at <https://beej.us/guide/bgipc/output/html/multipage/signals.html>.

Client

The client will take two (2) command line parameters: hostname and port number. The hostname and port number are **not optional** parameters. The following command will run the client program connecting to the server on port 12345.

```
./Client server_IP_address 12345  
(./Client IP_address port_number)
```

The online banking system is only available to registered clients of the online banking provider. Once the client program starts the client must authenticate by entering their user name and PIN which appears in the “*Authentication.txt*” file. Note that the PIN length may vary between clients. After the **server** validates the client’s credentials the client program can proceed to the main menu. If the client does not authenticate correctly the socket is to be immediately closed and the client program terminated. The server should still be running even if a client fails to authenticate. (See Figures 5 & 6)

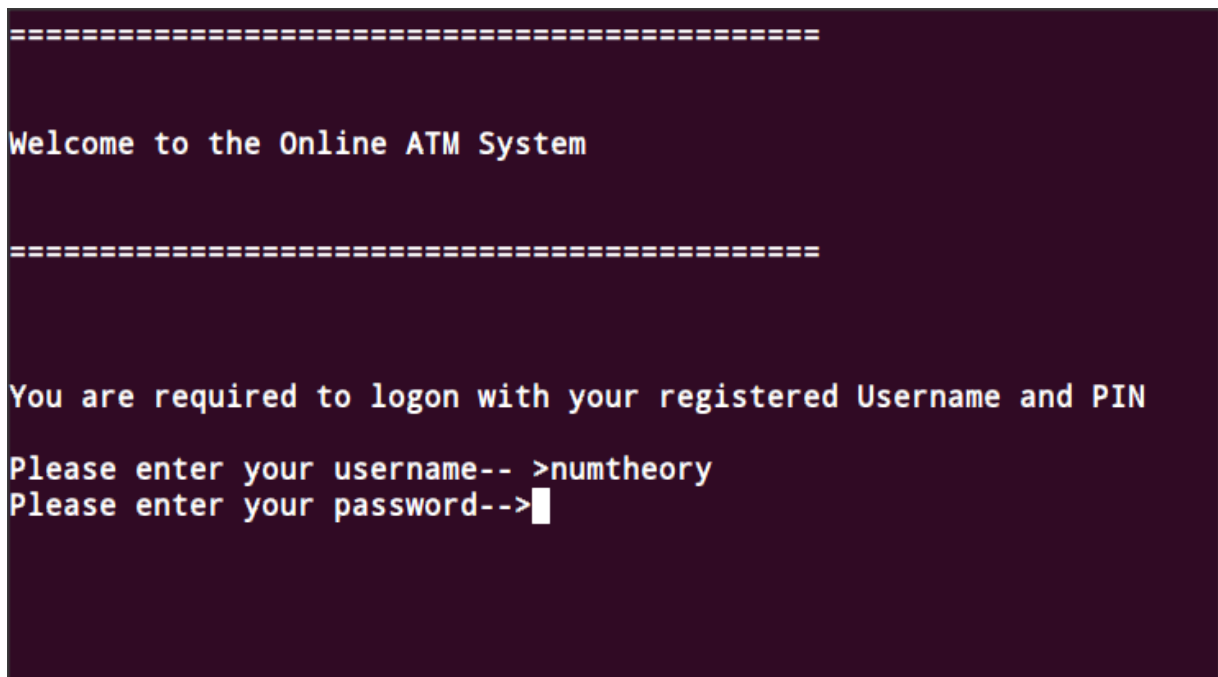


Figure 5 – Client logon screen

```
=====
Welcome to the Online ATM System
=====

You are required to logon with your registered Username and PIN

Please enter your username-- >numtheory
Please enter your password-->

You entered either an incorrect Username or PIN - disconnecting
anthony@GBVirtualBox:~/Documents/Anthony/Assignment_2_CAB403_2015$
```

Figure 6 – Unsuccessful Client logon

The client program should be a simple menu driven application. The client program will have six (6) options: *Account Balance*, *Withdrawal*, *Deposit*, *Transfer*, *Transaction Listing*, and *Exit*. (Figure 7)

When the client enters a selection option the user input must be validated. If the input is out of range or of an invalid type the client should be prompted to enter valid user input. (Figure 8).

```
Welcome to the ATM System

You are currently logged in as Carl Gauss
Client Number - 28510631

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->█
```


Figure 7 – Client Main Menu

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->gg

Invalid selection. Please select option from menu!

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->8

Invalid selection. Please select option from menu!

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->0

Invalid selection. Please select option from menu!
```

Figure 8 – Invalid user input

Account Balance – If the client elects to get an account balance the client must then select which account to query. Remember a client may have only a single account type or may have as many as three different account types. Once the client selects “Account Balance” from the menu the client program must then display only the account types owned by the client. Once the client selects the

account type the account balance for that account type is displayed. Figures 9 & 10.

```
Welcome to the ATM System

You are currently logged in as Carl Gauss
Client Number - 28510631

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Enter your selection (E/e to exit) -
```

Figure 9 – Client must select which account type to query for account balance

Only the account types owned by the client should be displayed as options to query for the account balance. If a client does not have for example a credit card account then this should *not* be an option shown to the client. Only those account types which are owned by the client should be displayed as options after selecting Account Balance from the main menu. Notice in Figure 11 that the client only has a Savings Account and a Loan Account so only two options are available to the client for Account balance.

NOTE: The current balance is the balance at the time of the client request and must take into account any previous transactions.

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Enter your selection (E/e to exit) - 1

=====

Account Name - Carl Gauss

Current Balance for Account 11012342 : $7815.16

=====
```

Figure 10 – Client elects to get account balance for Savings Account

```
You are currently logged in as Hertha Ayrton
Client Number - 64875412

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account

Enter your selection (E/e to exit) -
```

Figure 11 – Client only has 2 account types – Credit Card is *not* shown as an option.

If the request for an Account Balance is made at the time of server start up and no transactions have been completed then the current balance will be the Closing balance figure stored in “*Accounts.txt*”. If the request is made after the client has successfully completed transactions during the server instance then the current balance will be the sum of the closing balance figure stored in *Accounts.txt* plus any deposits, less any withdrawals and/or transfers. (See Figures 12, 13 & 14).

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Enter your selection (E/e to exit) - 3

=====

Account Name - Carl Gauss

Current Balance for Account 13014586 : $-165.22

=====
```

Figure 12 – Account Balance for client – Note the balance is the same as stored in *Accounts.txt*

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->2

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 2

Enter the amount to withdraw (E/e to exit) : $615.48

Withdrawal Completed: Closing Balance : $-780.70

=====
```

Figure 13 –Withdrawal from Credit Card account changes the account balance

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Enter your selection (E/e to exit) - 3

=====

Account Name - Carl Gauss

Current Balance for Account 13014586 : $-780.70

=====
```

Figure 14 – Account Balance reflects previous withdrawal from Credit Card account

Withdrawal – If the client elects to withdraw money then the accounts available for withdrawal that are owned by the client must be displayed. *Note that it is not possible to withdraw money from a Loan Account.* A withdrawal may only be made from a Savings account or from a Credit Card account and only if the client owns accounts of these types. Figure 15 shows a client with a Savings account, Credit Card account and a Loan Account however since withdrawal has been selected from the menu only the Savings Account and the Credit Card account are shown as options.

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Enter your selection (E/e to exit) - e

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->2

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - █
```

Figure 15 – Note client has a Savings, Loan and Credit Card account – Withdrawal is selected from menu and only Savings Account and Credit Card are displayed as account options

If the client elects to withdraw money from their Savings account the client is only able to withdraw an amount *equal to or less than* the current balance of the Savings account. If the client owns a Credit Card account and elects to withdraw money from their Credit Card they must stay within a \$-5000.00 limit. All credit cards in the ATM system have the same maximum limit of \$-5000.00.

The client should enter the amount to withdraw and the request must be processed on the *server side*. If there are insufficient funds the transaction should be declined. If the withdrawal is completed successfully the new account balance for the account must be displayed. (See Figure 16)

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->2

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 1

Enter the amount to withdraw (E/e to exit) : $65.29

Withdrawal Completed: Closing Balance : $7749.87

=====
```

Figure 16 – Successful withdrawal of funds from client Savings Account

A Savings Account must always maintain a positive (or \$0.00) balance. It is not possible to withdraw more than the current account balance of a Savings Account. Therefore it is not possible for a Savings account to have a negative balance. If there is insufficient funds available the transaction must be declined. (See Figure 17).

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->2

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 1

Enter the amount to withdraw (E/e to exit) : $8000

Insufficient Funds - Unable to process request

=====
```

Figure 17 – Attempt to withdraw amount greater than current account balance of Savings Account – Transaction must be declined

A Credit Card account may have a negative balance (money is owed to the virtual banking provider) or it may have a positive balance. The amount available to withdraw from a Credit Card account is determined by the formula:

$$\text{Available Funds} = 5000 + \text{Credit Card balance}$$

E.g. Credit Card Balance: \$-450.00

$$\text{Available Funds} = 5000 + -450 = 4500$$

E.g. Credit Card Balance: \$625.25

$$\text{Available Funds} = 5000 + 625.25 = 5625.25$$

As stated previously the maximum Credit Card limit is \$-5000.00. There is no maximum **positive** balance on a Credit Card account only a maximum **negative** balance. (See Figures 18, 19 & 20).


```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->1

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Enter your selection (E/e to exit) - 3

=====

Account Name - Carl Gauss

Current Balance for Account 13014586 : $-780.70

=====
```

Figure 18 – Current balance for Credit Card account for client

```
Selection option 1-6 ->2

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 2

Enter the amount to withdraw (E/e to exit) : $4500

Insufficient Funds - Unable to process request

=====
```

Figure 19 – Attempt to withdraw from Credit Card account will exceed \$5000 limit – Transaction declined

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->2

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 2

Enter the amount to withdraw (E/e to exit) : $4125.32

Withdrawal Completed: Closing Balance : $-4906.02

=====
```

Figure 20 – Withdrawal from Credit Card successful – Closing balance is less than \$-5000 limit

Deposit – If the client elects to deposit funds the client should display the accounts available to accept the deposit. All account types (Savings, Loan and Credit Card accounts may receive deposits). Only those account types owned by the client should be displayed. In Figure 21 the client only has a Savings account and a Credit Card account.

Clients must be warned that there is a maximum limit of \$1000.00 per deposit transaction. Note this is a transaction limit and *not* a daily/server instance limit. If the client wishes to deposit more than \$1000.00 the client will need to do multiple deposits. If the client tries to deposit more than \$1000.00 the client should be advised that the transaction cannot be processed and then be prompted to enter an amount of \$1000.00 or less. (See Figure 22).

Once the client has successfully deposited an amount of \$1000 or less the transaction should be processed on the server side and the new account balance displayed to the client. (See Figure 23).

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->3

=====

The maximum daily deposit is $1000.00!

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 
```

Figure 21 – The account type to receive the deposit must be selected if client selects Deposit from main menu – Only those account types owned by client should be displayed.

```
=====

The maximum daily deposit is $1000.00!

Select Account Type
1. Savings Account
2. Credit Card

Select Account Type (E/e to exit) - 1

Enter the amount to deposit (E/e to exit) : $1652.22

You cannot deposit more than $1000 in a single transaction!

Enter the amount to deposit (E/e to exit) : $
```

Figure 22 – Attempt to deposit more than \$1000 – Client is prompted to enter a smaller amount

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->3

=====

The maximum daily deposit is $1000.00!

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Select Account Type (E/e to exit) - 1

Enter the amount to deposit (E/e to exit) : $645.98

Deposit Completed: Closing Balance : $8395.85

=====
```

Figure 23 – Deposit successful – New account balance displayed to client for account.

Transfer – The ATM should be able to transfer money between registered client accounts of the online banking system. Clients may transfer money between accounts that they own or they may transfer money to an account owned by another client (referred to as an External account). It is not possible to transfer money from a client account to a non-existent account. The destination account must be a registered account in the “*Accounts.txt*” file.

Just as it is not possible to withdraw money *from* a Loan account it is *not possible* to transfer money *from* a Loan account. It is possible however to transfer money *to* a Loan account. Loan accounts may only accept money that is either directly deposited into the Loan account or transferred from a Savings or Credit Card account.

If a client elects to transfer money from the main menu the client should be shown the accounts from which money may be transferred **from** (either a Savings or Credit Card account). As with other transaction types only those applicable account types owned by the client should be displayed. In Figure 24 the client has all three account types (Savings, Loan and Credit Card accounts)

however the only options displayed from which to transfer money **from** are Savings account and Credit Card when they select Transfer from the main menu.

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->4

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account to Transfer From (E/e to exit) -
```

Figure 24 – Client prompted which account to transfer money *from*

After selecting which account to transfer money *from* the client is prompted for the destination account (i.e. account the money is to be transferred *to*). Figure 25.

```
Select Account Type
1. Savings Account
2. Credit Card

Select Account to Transfer From (E/e to exit) - 1

Select Account Type
1. Loan Account
2. Credit Card
3. External Account

Select Account to Transfer To (E/e to exit) - █
```

Figure 25 – Client prompted for the destination account (account to receive money)

In Figure 24 & 25 the client has elected to Transfer money from the main menu and then elected to transfer money *from their Savings Account*. The client then needs to select the destination account. Note that the account the client elects to transfer money from **cannot** be the same as the destination account. In Figure 25 the client elected to transfer money from their Savings account hence the Savings account does not appear as a destination account option. **The account**

which money is transferred from cannot be the same as the account which receives the money (the destination account).

The client is able to transfer money between accounts for which they own **or** they may elect to transfer money to an external account. An external account is an account owned by another client.

Internal Transfer – An internal transfer is defined as a transfer of funds between accounts which are owned by the client. If the client elects to do an internal transfer the client needs to enter the amount they wish to transfer. The amount the client can transfer is determined by the current balance of the account from which they are transferring money from. If for example they elect to transfer money from their Savings account the client may only transfer an amount equal to or less than the current balance of the Savings account.

If the client elects to transfer money from their Credit Card the amount they can transfer is determined by the same constraints imposed for Credit Card withdrawals – the client cannot exceed the \$5000 maximum limit. If for example the client has a Credit Card balance of \$-1000.00 the client is only able to transfer a maximum of \$4000 **from** their Credit Card account.

Remember it is ***not possible to transfer money from a Loan Account*** however the client may transfer money ***to a Loan Account***. Figure 26 shows an internal transfer from the client's Savings account to the client's Loan account. If there is sufficient funds the transaction should be processed and the account balances of both accounts updated. If both the ***from account and destination account*** are owned by the same client the new balances for **both** accounts should be displayed on the screen. The account balance of the *from account* is reduced by the amount transferred and the *destination account* balance is increased by the same amount. (Figure 26).

If there is not sufficient funds for the amount entered the client should be prompted to enter a valid amount to transfer. (Figure 27).

```
Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->4

=====

Select Account Type
1. Savings Account
2. Credit Card

Select Account to Transfer From (E/e to exit) - 1

Select Account Type
1. Loan Account
2. Credit Card
3. External Account

Select Account to Transfer To (E/e to exit) - 1

Enter the Amount to Transfer (E/e to exit) - $98.25

INTERNAL TRANSFER

Deducted $98.25 From: Account 11012342 - Closing Balance - $8229.14
Transfer $98.25 Dest: Account 12013464 - Closing Balance - $-158.53

=====
```

Figure 26 – Internal transfer – money transferred from Savings to Loan account

```
Select Account Type
1. Loan Account
2. Credit Card
3. External Account

Select Account to Transfer To (E/e to exit) - 2

Enter the Amount to Transfer (E/e to exit) - $98712.25

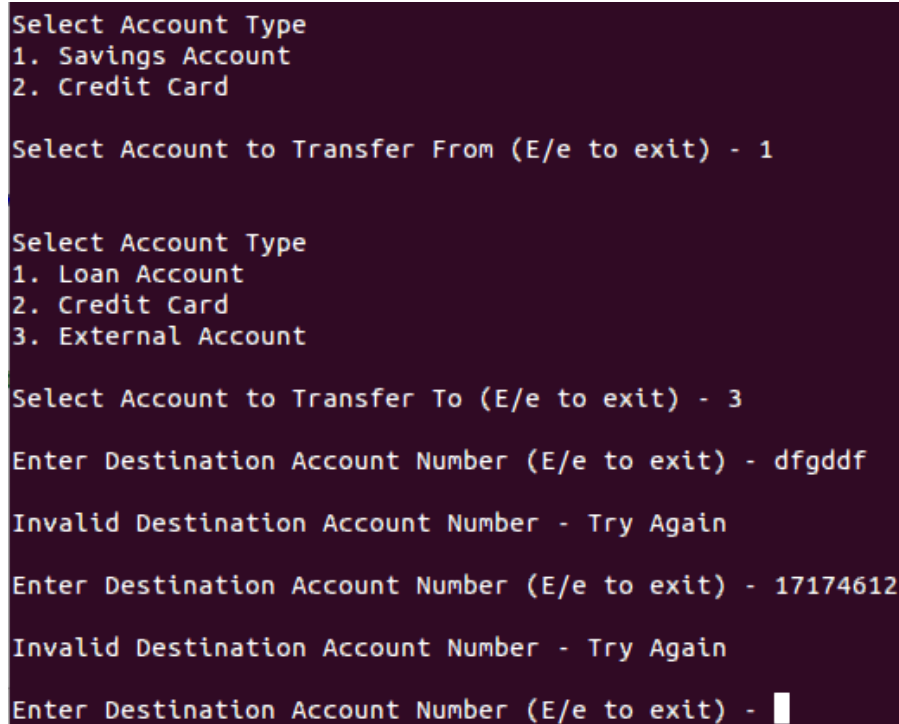
Invalid Amount - Enter a Valid Amount to Transfer.

Enter the Amount to Transfer (E/e to exit) - $
```

Figure 27 – Insufficient funds available to transfer – Client prompted to enter a valid amount

External Transfer – The client may elect to transfer money to an external account. An external account is an account **not owned** by the client. The external account selected must be a valid account which appears in the “*Accounts.txt*” file. The same constraints apply to the amount that a client may transfer to an external account as previously discussed for transfers to an internal account (an account owned by the client).

If the client elects to transfer money from an account they own to an external account the client must be prompted for the destination account number. If the account number exists and the account number is **not** an account owned by the client the transfer should be processed. If the client enters an account number which is an account owned by the client or the account does not exist the client should be prompted to enter a valid account. (Figure 28).



```
Select Account Type
1. Savings Account
2. Credit Card

Select Account to Transfer From (E/e to exit) - 1

Select Account Type
1. Loan Account
2. Credit Card
3. External Account

Select Account to Transfer To (E/e to exit) - 3

Enter Destination Account Number (E/e to exit) - dfgddf

Invalid Destination Account Number - Try Again

Enter Destination Account Number (E/e to exit) - 17174612

Invalid Destination Account Number - Try Again

Enter Destination Account Number (E/e to exit) - █
```

Figure 28 – Destination account entered is owned by client – Client prompted to re-enter a valid destination account (Note the invalid alpha account – Client prompted to try again)

Once the client enters a valid account number the client is then prompted for the amount to transfer. If there is sufficient funds the transfer is processed on the server and the new account balance of the *from* account is displayed however the destination account balance is **not displayed**. The destination account is an account owned by a different client therefore for privacy reasons the destination account balance is not shown. See Figure 29.

```
=====
Select Account Type
1. Savings Account
2. Credit Card

Select Account to Transfer From (E/e to exit) - 2

Select Account Type
1. Savings Account
2. Loan Account
3. External Account

Select Account to Transfer To (E/e to exit) - 3

Enter Destination Account Number (E/e to exit) - 15754794

Enter the Amount to Transfer (E/e to exit) - $412.85

EXTERNAL TRANSFER

Deducted $412.85 From: Account 17174612 - Closing Balance - $-397.61
Transfer $412.85 Dest: Account 15754794
=====
```

Figure 29 – Transfer from client Credit Card to an external account – The current balance of the Credit Card is displayed however the destination account balance is not shown.

Transaction Listing – A client must be able to request a full transaction history for any account for which they own. After a client has selected transaction listing from the main menu the client then selects the account type for which they require a transaction listing. The only account type options displayed are those account types owned by the client. If a client for example does not have a Loan Account however they have a Savings and Credit Card account then there should only be two account type options (Savings Account, Credit Card).

The transaction listing should display:

- The account owners full name
- The number of transactions for the selected account
- The Opening Balance of the account (stored in “*Accounts.txt*” file)
- All transactions the server has successfully processed for the selected client account
- Transaction type

See Figure 30 for a sample transaction history.

```

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->5

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Show transaction listing - Select Account Type (E/e to exit) - 1
=====

Account Owner - Leo Euler

Total Number of Transactions: 7

Opening Balance - Account 11012375 : $9825.23

=====
Transaction      Type      Amount
  1      Withdrawal    -85.26
  2      Deposit      812.46
  3      Deposit      47.56
  4      Withdrawal   -915.88
  5      Transfer    -98.66
  6      Transfer   -815.54
  7      Transfer    412.54

Closing Balance : $9182.45
=====

```

Figure 30 – Transaction listing for a client’s Savings Account

The transaction history must include every transaction processed in the current server instance and any transactions completed in *previous server instances*. Transactional history must persist between server instances.

No transactions for any other account should be displayed except those transactions associated with the selected account. Transactions for accounts not owned by the client definitely must not be displayed. The transactions displayed must be transactions the client has successfully submitted to the server whether the transaction occurred in the current server instance or in a previous server instance.

Transactions for a *single server instance* must be listed in the order in which they are submitted to the system. The first transaction successfully processed by the server should be displayed first and the most recent transaction for an account should be listed last. In Figure 30 transaction number seven (7) was the last transaction submitted to the server for this client's Savings account.

It is a requirement that all transactions are stored in a text file "*Transactions.txt*" when the server is shut down. When the server is restarted all previous transactions must be loaded by the server. Due to the limited data that is stored in this file it is **not** a requirement to maintain transactional order after the server is shut down and restarted. It is a requirement however that all transactions associated with an account persist when the server is restarted. (See Figure 31).

```
Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Show transaction listing - Select Account Type (E/e to exit) - 1

=====

Account Owner - Leo Euler

Total Number of Transactions: 7

Opening Balance - Account 11012375 : $9825.23

=====
Transaction      Type      Amount
1                Transfer    412.54
2                Withdrawal -85.26
3                Deposit     812.46
4                Deposit     47.56
5                Withdrawal -915.88
6                Transfer    -98.66
7                Transfer    -815.54

Closing Balance : $9182.45
=====
```

Figure 31 – Transaction listing for same account as in Figure 30 after server restarted

Notice in Figure 31 that all seven (7) transactions for the Savings account are displayed even though the server has been restarted however the order of the transactions is slightly different. It is a requirement that **all** transactions from every server instance are listed however the specific order may vary **iff** the server is restarted. However while the server is running transactional data must be

maintained in the same order in which the transaction was submitted to the server. It is only when the server is restarted that it is acceptable for the order of the transactions to vary.

When listing the transactions for an account deposits should be displayed as positive amounts. Withdrawals and transfers *from* the client account should be displayed as *negative amounts*. Transfers into a client account (the destination account) should be displayed as a *positive amount*. See Figure 31:

- Deposits are shown as a positive value
- Withdrawals are shown as a negative value
- Transfers *to* the account are shown as a positive value (Transaction 1)
- Transfers *from* the account are shown as a negative value (Transaction 7)

If a client elects to transfer money from one account to another the end result is two (2) transactions are generated. In Figure 29 an amount of \$412.85 is transferred from account 17174612 to account 15754794 therefore a transfer must appear in both of these accounts. The amount transferred will be shown as a negative amount in the transaction listing for account 17174612 (*from account*) and as a positive amount in the transaction listing for account 15754794 (*destination account*). See Figures 32 & 33.

```

=====
Account Owner - George Boole
Total Number of Transactions: 3
Opening Balance - Account 17174612 : $0.00
=====
  Transaction      Type      Amount
      1           Transfer    -412.85
      2           Withdrawal   -94.65
      3             Deposit    447.98
Closing Balance : $-59.52
=====

```

Figure 32 – Account 17174612 transaction listing – Transfer shown as a negative value

If the client requests a transaction listing for an account which has no recorded transactions the client should receive a simple narration reporting there are no transactions on file for the chosen account. (Figure 34)

```
=====
Account Owner - Mary Cartwright
Total Number of Transactions: 1
Opening Balance - Account 15754794 : $9254.22
=====
Transaction      Type      Amount
      1      Transfer      412.85
Closing Balance : $9667.07
=====
```

Figure 33 – Account 15754794 transaction listing – Transfer shown as positive value

The transaction listing should **not** include any transactions that were declined by the server. Only transactions that have been successfully processed by the server for the chosen account must be displayed.

```
=====
Account Owner - George Boole
Total Number of Transactions: 0
Opening Balance - Account 14532364 : $22350.15
There are no transactions on file for this account
Closing Balance : $22350.15
=====
```

Figure 34 – No transactions for the selected account – Opening & Closing balance are equal

Additional Functionality - It is important to validate user input for *all transaction types*. If the client inputs a non-numeric entry, a negative value or a value of zero (0) the client should be alerted that the input is invalid and prompted to enter a valid amount. See Figure 35.

```
The maximum daily deposit is $1000.00!

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Select Account Type (E/e to exit) - 1

Enter the amount to deposit (E/e to exit) : $gg

You have entered an invalid amount - Enter an amount of $1000 or less!

Enter the amount to deposit (E/e to exit) : $0

You cannot enter amounts of $0.00 or less!

Enter the amount to deposit (E/e to exit) : $-56.23

You have entered an invalid amount - Enter an amount of $1000 or less!

Enter the amount to deposit (E/e to exit) : $18.22

Deposit Completed: Closing Balance : $8611.72
```

Figure 35 – Examples of invalid deposit amounts – input must be validated

In addition the client also should be given the option to cancel a transaction. If the client elects to withdraw, deposit, transfer funds or get a transaction listing there must be an option for the client to cancel the transaction and not proceed with the chosen option. The client must be able to exit the transaction at any stage during the process. How this is accomplished is an implementation and design issue. The option to exit a transaction using “E/e to exit” as shown in the example screen shots is a suggestion only. The exact method implemented to exit a transaction process is left to the programmer’s discretion however it should be simple and easy to use for the client. Cumbersome methods to exit a transaction process or forcing the client to complete the transaction will incur a penalty against usability. See Figure 36.


```
Select Account Type
1. Savings Account
2. Credit Card

Select Account to Transfer From (E/e to exit) - 1

Select Account Type
1. Loan Account
2. Credit Card
3. External Account

Select Account to Transfer To (E/e to exit) - 3

Enter Destination Account Number (E/e to exit) - 15754794

Enter the Amount to Transfer (E/e to exit) - $e
```

Figure 36 – Client should be given an option to exit a transaction process at any stage.

Exit Program - The final explicit functionality required in Task 1 is to permit the client to exit the ATM. If the client elects to disconnect from the server the client program should exit gracefully closing the open socket and free any dynamically allocated memory. Once the client exits another client should be able to log into the server. In Task 1 only one client at a time is required to be able to log into the server. It is not a requirement in Task 1 for there to be multiple concurrent client connections to the server.

Remember that all transaction history must be maintained during and between server instances. If a client logs out of the server having successfully completed any number of transactions, when that client logs back into the server those previous transactions must still be displayed if a transaction listing is requested. Figure 37 shows the transaction listing for Account 11012342 *after* logging out of the server and then at a later time logging back into the server. Notice that all the previous successful transactions are listed even though the client has just reconnected to the server for a second time and has immediately requested a transaction listing. Transactions must be stored on the server side to permit the client to log out and then log back in and still be able to see all the transactions completed from previous client/server sessions.

```
Welcome to the ATM System

You are currently logged in as Carl Gauss
Client Number - 28510631

Please enter a selection
<1> Account Balance
<2> Withdrawal
<3> Deposit
<4> Transfer
<5> Transaction Listing
<6> EXIT

Selection option 1-6 ->5

Select Account Type
1. Savings Account
2. Loan Account
3. Credit Card

Show transaction listing - Select Account Type (E/e to exit) - 1
=====

Account Owner - Carl Gauss

Total Number of Transactions: 4

Opening Balance - Account 11012342 : $7815.16

=====
Transaction      Type      Amount
   1      Withdrawal    -65.29
   2      Deposit      645.98
   3      Deposit      197.65
   4      Deposit       18.22

Closing Balance : $8611.72
=====
```

Figure 37 – Client has logged in and immediately requested a transaction list – all transactions from previous client/server session are stored on the server side.

The required explicit functionality for Task 1 in summary:

1. Account Balance
2. Deposit
3. Withdrawal
4. Transfer
5. Transaction Listing
6. Exit from server

Implicit Server Functionality

It is a requirement that the server maintains all transactional data for every client that logs into the ATM. This includes transactions that were completed in a previous server instance. This requires your implementation to maintain two (2) of the supplied files:

1. Accounts.txt
2. Transactions.txt

1. Accounts.txt

The only data that needs to be updated in “*Accounts.txt*” is the Closing Balance. Your implementation needs to keep a running closing balance to ensure that the account data requested by the client is current. When the server starts the information in “*Accounts.txt*” needs to be tokenised and stored in memory. The closing balance may be updated in volatile memory during server runtime however when the server receives an interrupt signal to shut down the closing balance for all the clients must be updated to reflect the true closing balance for each client account. This means that when the server is restarted the closing balance for the client reflects the true amount of available funds just as it would in any online banking system. It is required that the Account Number order in “*Accounts.txt*” is maintained. The account numbers must not be changed in the file and the order in which the account numbers appear must remain the same. The only data to be changed in this file is the closing balance (value stored in the ClosingBal column).

Figure 38 shows the data contained in “*Accounts.txt*” when the server runs for the **first time** (no transactions have been completed by any client). Note that the amounts in the Opening Balance column (OpeningBal) are the same as the amounts in the Closing Balance column (ClosingBal).

AccountNo	OpeningBal	ClosingBal
11012342	7815.16	7815.16
12013464	-256.78	-256.78
13014586	-165.22	-165.22
11012375	9825.23	9825.23
12013500	-10022.15	-10022.15
13014625	-3598.25	-3598.25
11145442	982.05	982.05
12158664	-988.21	-988.21
11174581	12.20	12.20
12190452	-100.25	-100.25
13206323	-4825.26	-4825.26
13446565	1995.26	1995.26
14668980	-65.21	-65.21
15891395	-1987.25	-1987.25
14532364	22350.15	22350.15
15853488	-2654.25	-2654.25
17174612	0.00	0.00
14532419	1987.12	1987.12
14544365	84.23	84.23
17188795	0.00	0.00
17053500	-4554.12	-4554.12
18474625	-365.24	-365.24
15754794	9254.22	9254.22
18619302	-987.25	-987.25

Figure 38 – Accounts.txt before any transactions have been completed by clients

Figure 39 shows a sample of “*Accounts.txt*” file after clients have logged into the server, successfully completed transactions on some of their accounts and the server subsequently shut down. Notice that Carl Gauss has three (3) accounts (Savings Account, a Loan Account and a Credit Card account). Account Number 11012342 (Carl Gauss Savings Account) has a closing balance of \$7622.24. This is the balance shown in Figure 40 after three (3) transactions have been processed against this clients Savings Account.

AccountNo	OpeningBal	ClosingBal
11012342	7815.16	7622.24
12013464	-256.78	-57.53
13014586	-165.22	-1020.69
11012375	9825.23	9825.23
12013500	-10022.15	-10022.15

Figure 39 – “Accounts.txt” file after clients have transacted on their accounts and the server shut down. Notice the opening balance is unchanged however the closing balance for some accounts has changed

```

=====
Account Owner - Carl Gauss
Total Number of Transactions: 3
Opening Balance - Account 11012342 : $7815.16
=====
Transaction      Type      Amount
    1      Withdrawal    -187.25
    2      Withdrawal    -94.12
    3          Deposit     88.45
Closing Balance : $7622.24
=====

```

Figure 40 – Transaction listing for account 11012342

The closing balance in Figure 40 is the same as the closing balance stored in “*Accounts.txt*” file as is the opening balance (Figure 39). The transaction listing shows the transactions that have processed on account 11012342. After this listing was generated by the client the server was shut down. Your implementation must update the closing balance (amount in ClosingBal column) when the server receives an interrupt signal.

After the server shuts down the closing balance in “*Accounts.txt*” file (ClosingBal) must reflect the true closing balance for the client.

NOTE: It is recommended that you keep a backup copy of *Accounts.txt* while you are developing your solution.

2. Transactions.txt

Every transaction that is successfully completed must be able to be retrieved by the ATM. A full transactional history is required to be maintained for every client account. This includes transactions that were submitted by clients in previous server instances. While transactions during a server instance may be maintained in volatile memory a method to maintain transactions from previous server instances requires persistent storage. “*Transactions.txt*” can be used for this purpose.

When the server starts for the first time “*Transactions.txt*” will not contain any transactions. As clients transact on their accounts transactions stored in volatile memory must eventually be written to persistent storage. This will

permit clients to obtain a full historical listing of all their transactions even if the server is shut down and restarted. “*Transactions.txt*” file should be used to store all account transactions. When the server is restarted it should load all the transactions stored in this file to enable clients to obtain a full historical transaction listing for all their accounts from every server instance.

Transactions must be stored in ascending *from account number order*.

“*Transactions.txt*” needs to maintain four (4) attributes for each transaction:

- From Account Number (FromAccount column)
- To Account Number (ToAccount column)
- Transaction Type (2=Withdrawal, 3= Deposit, 4 = Transfer)
- Amount (appropriately signed as negative or positive)

Figure 41 shows the data as it may appear after the server has shut down for the first time.

FromAccount	ToAccount	TranType	Amount
11012342	11012342	2	-187.25
11012342	11012342	2	-94.12
11012342	11012342	3	88.45
11145442	11145442	3	955.54
11145442	12158664	4	-94.54
12013464	12013464	3	199.25
12158664	12158664	3	999.15
13014586	15754794	4	-855.47

Figure 41 – Example Transactions.txt file as it may appear after server shut down

Figure 41 has three (3) transactions for account 11012342. These are the transactions shown in Figure 40 and consists of two (2) withdrawals and one (1) deposit. Notice that the withdrawals (TranType = 2) are negative amounts and deposits (TranType = 3) are positive amounts. For both deposits and withdrawals the account numbers stored in the FromAccount and ToAccount columns are *always* the same.

Notice that for transfers (TranType = 4) the amount is *always negative* however the account number in the FromAccount column is *always* different to the account number in the ToAccount column. Account 11145442 has two (2) transactions stored in “*Transactions.txt*” – one (1) deposit (TranType = 3) and one (1) transfer (TranType = 4). The transaction listing for this account is shown in Figure 42.

```

=====
Account Owner - Hertha Ayrton
Total Number of Transactions: 2
Opening Balance - Account 11145442 : $982.05
=====
Transaction      Type      Amount
    1           Deposit      955.54
    2           Transfer     -94.54
Closing Balance : $1843.05
=====

```

Figure 42 – Transaction listing for account 11145442

The transfer stored in “*Transactions.txt*” (Figure 41) in the amount of \$-94.54 has account number 11145442 stored in the FromAccount column. This is the transfer shown in Figure 42. As stated this is always a negative amount in the transaction listing.

The account number stored in the ToAccount column for this transfer is 12158664. This transaction is shown in the transaction listing in Figure 43. Note that this amount is shown as positive.

```

=====
Account Owner - Hertha Ayrton
Total Number of Transactions: 2
Opening Balance - Account 12158664 : $-988.21
=====
Transaction      Type      Amount
    1           Transfer      94.54
    2           Deposit     999.15
Closing Balance : $105.48
=====

```

Figure 43 – Transaction listing for account 12158664

This single transfer transaction listing in “*Transactions.txt*” must be able to generate the transfer transaction for both transaction listings. A transfer transaction is recorded only **once** in “*Transactions.txt*” however your implementation must use this single transaction entry to generate both transactions. Do not record transactions of type *transfer* twice. Using the from

account number and to account number your implementation must be able to generate both transactions. While this transfer transaction was an internal transfer (transfer between two accounts owned by the same client) the same applies for external transfers (transfer from account owned by one client to an account owned by a different client).

The transfer of \$-855.47 in Figure 41 similarly must generate two transactions. In this case the account number stored in the FromAccount column is 13014586 and the account number stored in the ToAccount column is 15754794. Figure 44 shows the transaction listing for account 13014586. Account 13014586 is stored in the FromAccount column therefore the amount is shown in the transaction listing for this account as a negative value.

```
=====
Account Owner - Carl Gauss
Total Number of Transactions: 1
Opening Balance - Account 13014586 : $-165.22
=====
Transaction      Type      Amount
      1          Transfer    -855.47
Closing Balance : $-1020.69
=====
```

Figure 44 – Transaction listing for account 13014586

Figure 45 shows the transaction listing for account 15754794. This account number is stored in the ToAccount column in “*Transactions.txt*” therefore the amount shown in the transaction listing is a positive value.

Your implementation must be able to use a single transfer transaction entry in the “*Transactions.txt*” file to generate transaction listings for both the account from which the money was transferred from as well as the transaction listing for the destination account (the account that the money was transferred to).

When implementing your solution ensure consideration is given to minimising the overhead of file IO.

```
=====
Account Owner - Mary Cartwright
Total Number of Transactions: 1
Opening Balance - Account 15754794 : $9254.22
=====
Transaction      Type      Amount
      1      Transfer      855.47
Closing Balance : $10109.69
=====
```

Figure 45 – Transaction listing for account 15754794

The implementation for Task 1 does not need to handle concurrent connections and therefore does not need to be multithreaded. Only one client at a time can connect to the server. As such there is no requirement for synchronisation primitives.

The client/server system is all console based. No bonuses will be given for complex user interfaces however cumbersome user interfaces will be penalised.

Task 2:

You only need to attempt this task if you are aiming to achieve a mark up to 84% for this assignment.

The server you have implemented in Task 1 can only handle a single request at a time. You are required to extend the server to accept multiple concurrent connections. As each new connection is made a new thread is created to handle the client request.

The server will allow up to 10 clients to use the ATM at the same time. You will need to implement a system whereby the server can accept multiple concurrent connections. After the maximum of 10 clients have connected if a new client tries to connect either the new client will have to wait for another client to exit or the connection may be dropped. When a client exits the thread allocated to the client is destroyed.

NOTE: In Task 2 do not fork the parent process. If you attempt Task 2 you must use POSIX Threads.

Task 3:

You only need to attempt this task if you are aiming to achieve a mark up to 100% for this assignment.

The server will allow up to 10 clients to use the system at the same time. You will need to implement a thread pool where each incoming connection is handled in a separate thread. A thread pool reduces the cost of constantly creating and destroying threads to handle requests. Instead, threads are reused to handle connections. It also limits the number of incoming connections to guarantee the performance of the server.

You can implement your own Thread Pool reusing the producer consumer pattern from the weekly practicals, where the main thread accepts incoming sockets and places them in a queue (the producer) and 10 threads remove sockets from the queue and process the connection (the consumer). You may also use or adapt the complete Thread Pool implementation provided in the weekly practicals.

2 Submission

Your assignment solution will be submitted electronically via Blackboard on or before the due date. If you work in a group, you only need to submit one copy. Your submission should be a zip file and the file name should contain the student number of the contributors and include the following items:

- All source code of your solution.
- **README.txt** file with instructions on how to compile and run your program. Make sure you include instructions for compiling at the command line even if you supply a make file.
- The *make* file that you used for generating the executable file, however this is not mandatory.
- A marking criteria sheet with student names and student numbers
- **A report in Word Document Format or PDF which includes:**

- a) A statement of completeness indicating the tasks you attempted, any deviations from the assignment specification, and problems and deficiencies in your solution. State explicitly which task numbers were attempted and what functionality was attempted. If all functionality is implemented it is sufficient to state this explicitly.
- b) Description of the synchronisation primitives used to protect the data structures used in your implementation. Synchronisation primitives are essential if you attempt Task 2 or Task 3.
- c) Briefly discuss the data structures used to store data in your implementation. To successfully implement a solution for the ATM you will need to carefully consider which data structures should be used. Discuss the data structures that were chosen and the advantages and/or disadvantages of the chosen data structures.
- d) Information about your team, including student names and student numbers, if applicable.
- e) Statement of each student's contribution to the assignment. If all members of the group contributed equally then it is sufficient to state this explicitly.

Your report should be no longer than one (1) A4 page.

Note: If the program does not compile on the command line in the Linux environment used during each weekly practical your submission will be heavily penalised. Implementations written in Visual Studio or other IDE's that do not compile on the Linux command line will receive a mark of zero (0).

The original text files available on Blackboard will be used to assess your submission. It is essential that your implementation will work with these files in the format supplied on Blackboard and not a modified version of these files.

Late submission and plagiarism policies: QUT's standard policy for late submissions and plagiarism will apply. Information on these policies can be found at the following links:

- <https://www.student.qut.edu.au/studying/assessment/late-assignments-and-extensions>
- http://www.mopp.qut.edu.au/C/C_05_03.jsp
- http://www.mopp.qut.edu.au/E/E_08_01.jsp#E_08_01.08.mdoc