Bautista, Glen Angelo D.
C131

June 29, 2023
Prof. Maria Corazon Adriano

ATM v7.2 Final Version

Table of Contents

List of (Unique) Features

1. Complete error checking for all inputs
2. File structures (saved as .dat)
    a. Accounts
    b. Transactions
3. Passwords are encrypted in .dat file with custom key encryption algorithm
4. Admin Panel
    a. Create/Delete accounts
    b. Edit accounts
    c. List accounts (w/ Page support)
5. Transaction Log (w/ Page support)
    a. Global
    b. Per account
6. Accounts get locked if exceeding attempts
7. Transfer money
8. Colored text prompts

User Manual

How to use (simplified)

1. Run the program Bautista_ATMv7.2.exe.
2. Enter your account number and PIN.
3. Select a transaction from the menu.
4. Follow the instructions on the screen.
5. To exit the program, select option ‘0’ from the menu or type ‘n’ when prompted to do another transaction.

Login (user)

To login to your account, enter your account number and PIN.

Here are some sample accounts for you to try:

Name	**Account Number**	**PIN**
Glen	1001	0000
Angelo	1002	1234
Yui	1003	2468

Note: There are more accounts in the sample files, which can be accessed by logging in to the **Admin Panel > List Accounts**

Login (admin)

To login to the admin panel, enter 1000 as the account number and "admin" as the password.

Available Functions (user)

- Balance inquiry: This transaction displays your account balance.
- Deposit: This transaction adds money to your account balance.
- Withdrawal: This transaction deducts money from your account balance.
- Transfer money: This transaction transfers money from your account to another account.

- Transaction history: This transaction displays a list of all your recent transactions.
- Generate report: This transaction generates a report of all your transactions.
- Logout: This transaction logs you out of your account.

Available Functions (admin)

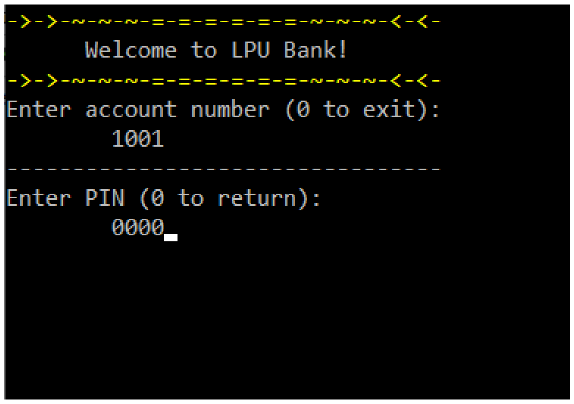
The admin panel allows you to manage all the accounts in the system.

- List accounts: This task displays a list of all the accounts in the system.
- Create account: This task creates a new account.
- Edit account: This task edits an existing account.
    - View details: Displays the details of the selected account, including the account number, name, balance, deposit count, withdrawal count, total deposit, and total withdrawal.
    - Change name: Change the name of the selected account.
    - Change PIN: Change the PIN of the selected account.
    - Lock/Unlock: Locks or unlocks the selected account
    - Delete Account: Deletes the selected account
    - Change Account Selection: Change the selected acount
- List all transactions: This task displays a list of all the transactions in the system.

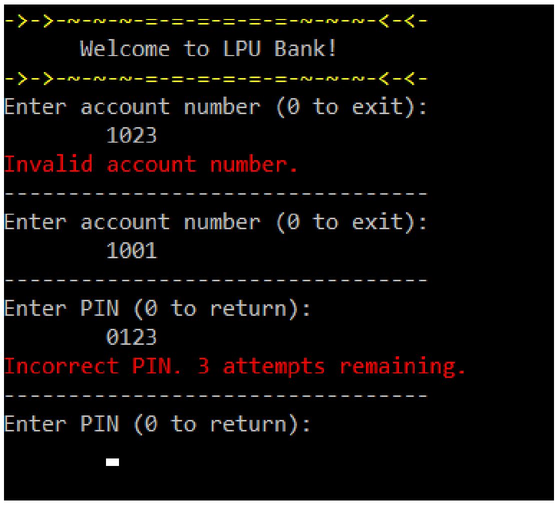
Common Error Messages

- Invalid account number: This error message is displayed if you enter an invalid account number.
- Invalid PIN: This error message is displayed if you enter an invalid PIN.
- Deposit amount should be greater than zero: This error message is displayed if you enter a negative deposit amount.
- Withdrawal amount exceeds balance: This error message is displayed if you try to withdraw an amount that exceeds your balance.
- Account is locked: This error message is displayed if you try to login to an account that is locked. Accounts can be unlocked with the admin panel.

Sample Output

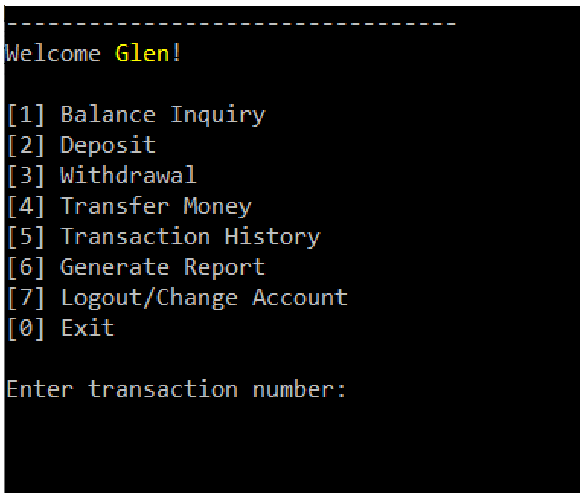


```
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
      Welcome to LPU Bank!
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
Enter account number (0 to exit):
        1001
---------------------------------
Enter PIN (0 to return):
        0000
```



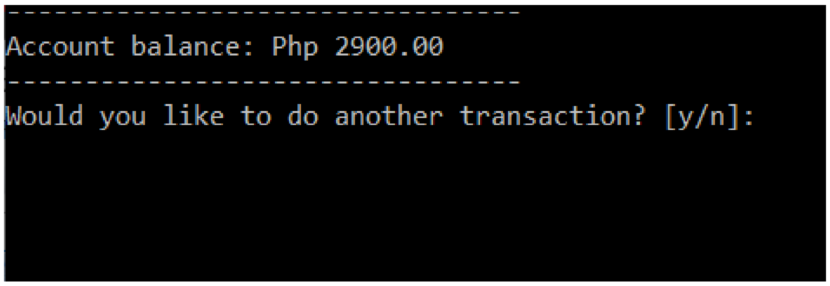
```
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
      Welcome to LPU Bank!
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
Enter account number (0 to exit):
        1023
Invalid account number.
---------------------------------
Enter account number (0 to exit):
        1001
---------------------------------
Enter PIN (0 to return):
        0123
Incorrect PIN. 3 attempts remaining.
---------------------------------
Enter PIN (0 to return):
```

Login menu: valid and invalid inputs



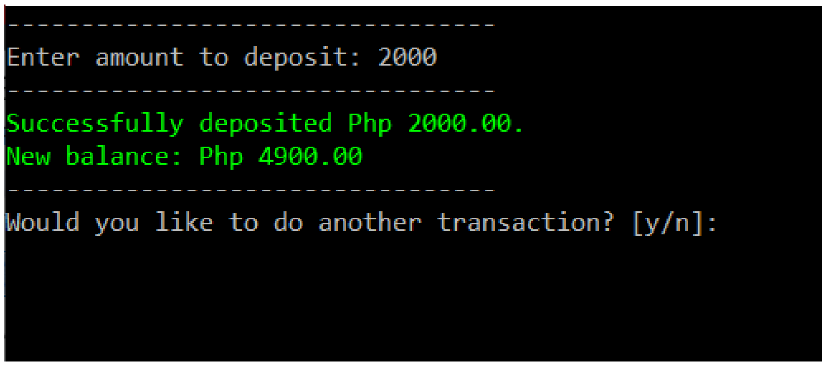
```
---------------------------------
Welcome Glen!

[1] Balance Inquiry
[2] Deposit
[3] Withdrawal
[4] Transfer Money
[5] Transaction History
[6] Generate Report
[7] Logout/Change Account
[0] Exit

Enter transaction number:
```

Transaction Menu (Account: 1001)



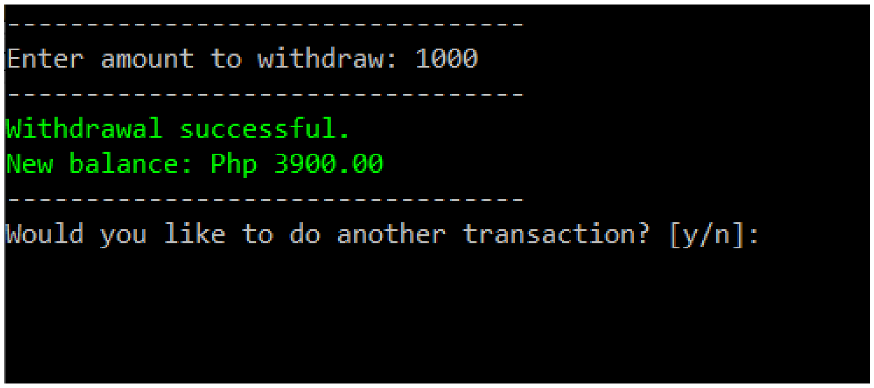
```
---------------------------------
Account balance: Php 2900.00
---------------------------------
Would you like to do another transaction? [y/n]:
```

[1] Balance Inquiry



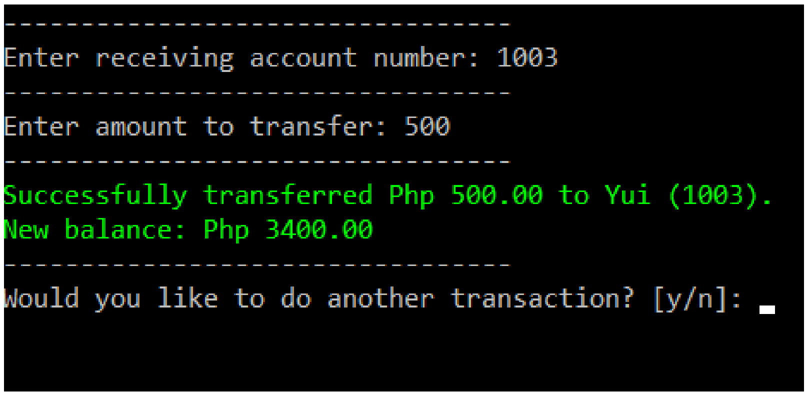
```
----------------------------------
Enter amount to deposit: 2000
----------------------------------
Successfully deposited Php 2000.00.
New balance: Php 4900.00
----------------------------------
Would you like to do another transaction? [y/n]:
```

[2] Deposit



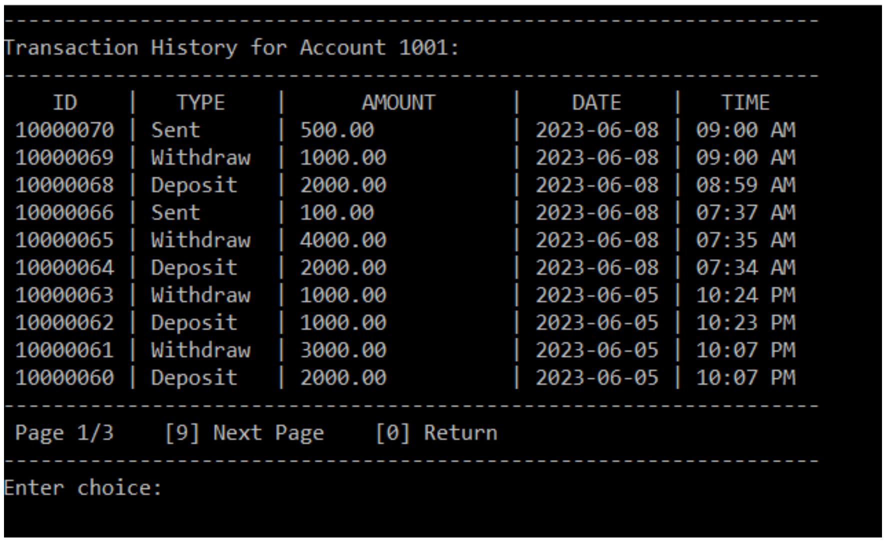
```
----------------------------------
Enter amount to withdraw: 1000
----------------------------------
Withdrawal successful.
New balance: Php 3900.00
----------------------------------
Would you like to do another transaction? [y/n]:
```

[3] Withdrawal

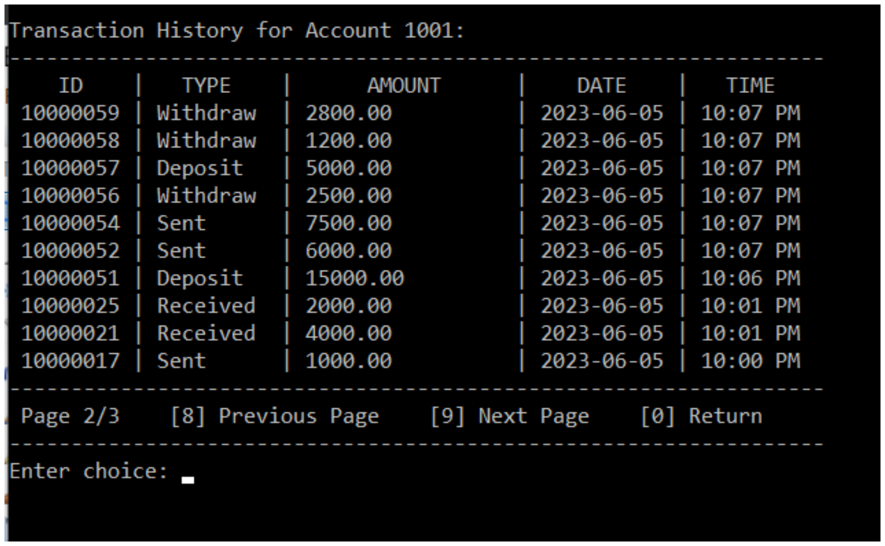


```
----------------------------------
Enter receiving account number: 1003
----------------------------------
Enter amount to transfer: 500
----------------------------------
Successfully transferred Php 500.00 to Yui (1003).
New balance: Php 3400.00
----------------------------------
Would you like to do another transaction? [y/n]:
```

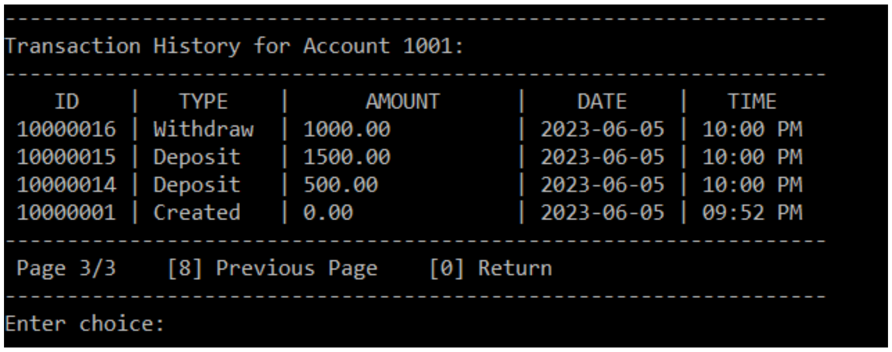
[4] Transfer Money



```
-----------------------------------------------------------------
Transaction History for Account 1001:
-----------------------------------------------------------------
    ID     |   TYPE    |       AMOUNT       |    DATE    |   TIME
 10000070 | Sent      | 500.00             | 2023-06-08 | 09:00 AM
 10000069 | Withdraw  | 1000.00            | 2023-06-08 | 09:00 AM
 10000068 | Deposit   | 2000.00            | 2023-06-08 | 08:59 AM
 10000066 | Sent      | 100.00             | 2023-06-08 | 07:37 AM
 10000065 | Withdraw  | 4000.00            | 2023-06-08 | 07:35 AM
 10000064 | Deposit   | 2000.00            | 2023-06-08 | 07:34 AM
 10000063 | Withdraw  | 1000.00            | 2023-06-05 | 10:24 PM
 10000062 | Deposit   | 1000.00            | 2023-06-05 | 10:23 PM
 10000061 | Withdraw  | 3000.00            | 2023-06-05 | 10:07 PM
 10000060 | Deposit   | 2000.00            | 2023-06-05 | 10:07 PM
-----------------------------------------------------------------
 Page 1/3    [9] Next Page    [0] Return
-----------------------------------------------------------------
Enter choice:
```

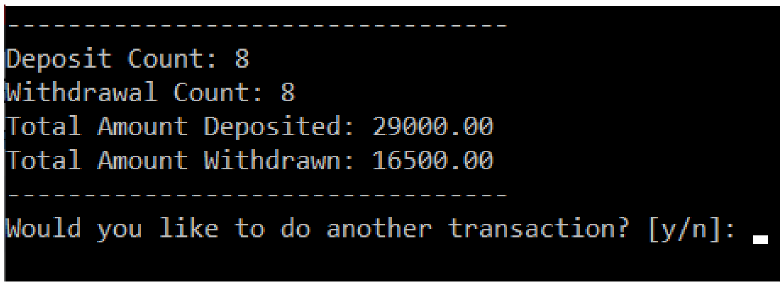


```
Transaction History for Account 1001:
-----------------------------------------------------------------
    ID     |   TYPE    |       AMOUNT       |    DATE    |   TIME
 10000059 | Withdraw  | 2800.00            | 2023-06-05 | 10:07 PM
 10000058 | Withdraw  | 1200.00            | 2023-06-05 | 10:07 PM
 10000057 | Deposit   | 5000.00            | 2023-06-05 | 10:07 PM
 10000056 | Withdraw  | 2500.00            | 2023-06-05 | 10:07 PM
 10000054 | Sent      | 7500.00            | 2023-06-05 | 10:07 PM
 10000052 | Sent      | 6000.00            | 2023-06-05 | 10:07 PM
 10000051 | Deposit   | 15000.00           | 2023-06-05 | 10:06 PM
 10000025 | Received  | 2000.00            | 2023-06-05 | 10:01 PM
 10000021 | Received  | 4000.00            | 2023-06-05 | 10:01 PM
 10000017 | Sent      | 1000.00            | 2023-06-05 | 10:00 PM
-----------------------------------------------------------------
 Page 2/3    [8] Previous Page    [9] Next Page    [0] Return
-----------------------------------------------------------------
Enter choice:
```



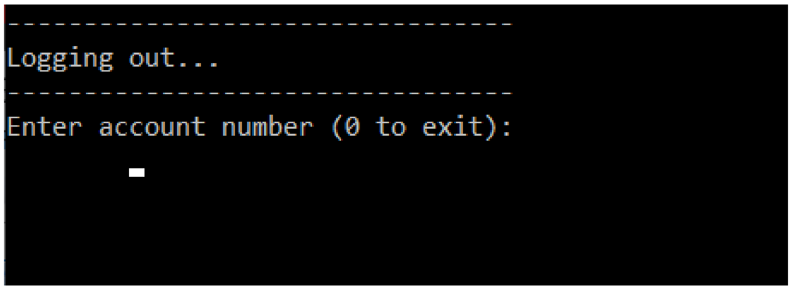
```
-----------------------------------------------------------------
Transaction History for Account 1001:
-----------------------------------------------------------------
    ID     |   TYPE    |       AMOUNT       |    DATE    |   TIME
 10000016 | Withdraw  | 1000.00            | 2023-06-05 | 10:00 PM
 10000015 | Deposit   | 1500.00            | 2023-06-05 | 10:00 PM
 10000014 | Deposit   | 500.00             | 2023-06-05 | 10:00 PM
 10000001 | Created   | 0.00               | 2023-06-05 | 09:52 PM
-----------------------------------------------------------------
 Page 3/3    [8] Previous Page    [0] Return
-----------------------------------------------------------------
Enter choice:
```

[5] Transaction History (Page 1, 2, 3)
Note that the deposit, withdraw, and transfer transactions we did are listed first in the transaction history.



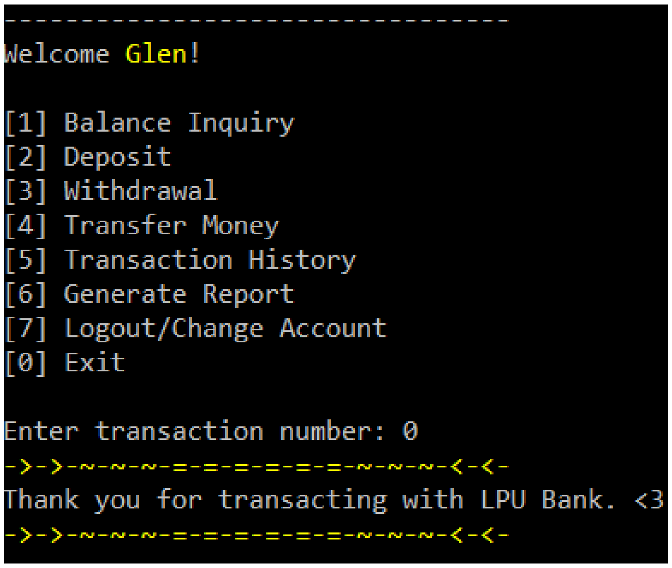
```
--------------------------------
Deposit Count: 8
Withdrawal Count: 8
Total Amount Deposited: 29000.00
Total Amount Withdrawn: 16500.00
--------------------------------
Would you like to do another transaction? [y/n]:
```

[6] Generate Report



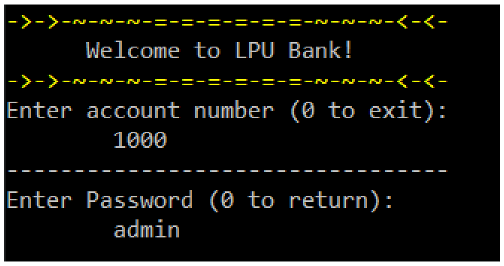
```
--------------------------------
Logging out...
--------------------------------
Enter account number (0 to exit):
```

[7] Logout / Change Account



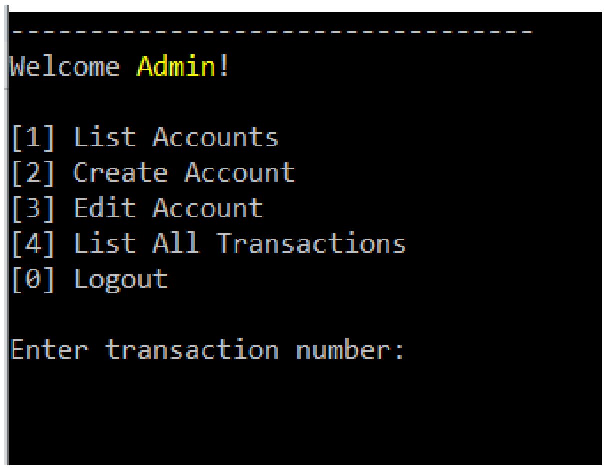
```
--------------------------------
Welcome Glen!

[1] Balance Inquiry
[2] Deposit
[3] Withdrawal
[4] Transfer Money
[5] Transaction History
[6] Generate Report
[7] Logout/Change Account
[0] Exit

Enter transaction number: 0
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
Thank you for transacting with LPU Bank. <3
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
```

[0] Exit



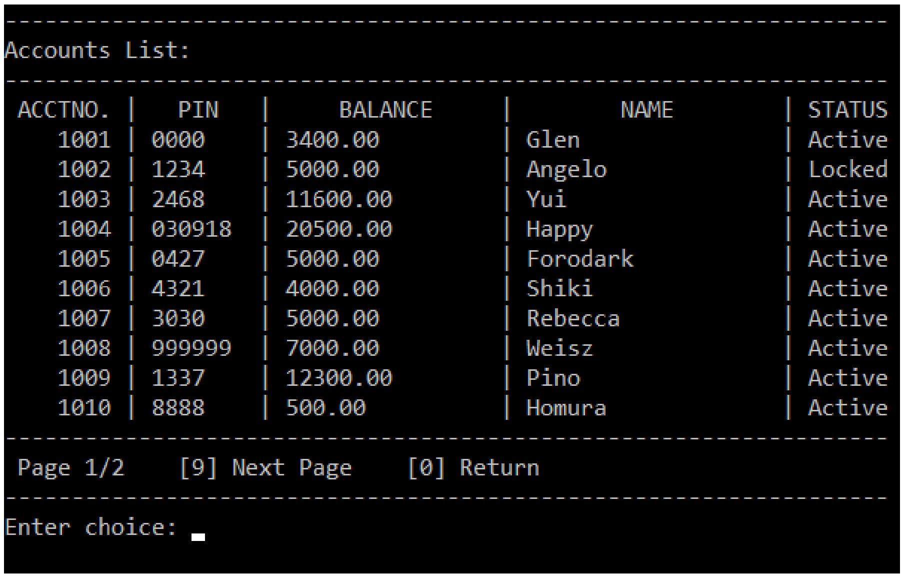
```
->->-~-~-~-=-=-=-=-=-=-=-~-~-~-<-<-
      Welcome to LPU Bank!
->->-~-~-~-=-=-=-=-=-=-=-~-~-~-<-<-
Enter account number (0 to exit):
        1000
---------------------------------
Enter Password (0 to return):
        admin
```

Login Menu: Admin account

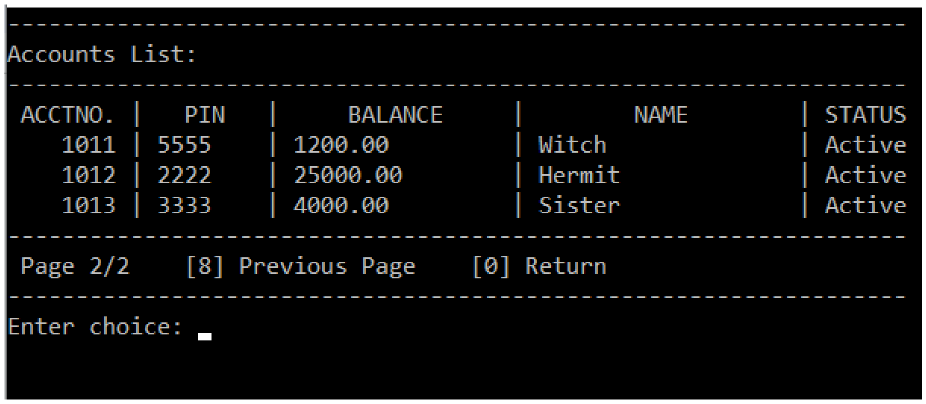


```
----------------------------------
Welcome Admin!

[1] List Accounts
[2] Create Account
[3] Edit Account
[4] List All Transactions
[0] Logout

Enter transaction number:
```

Admin Panel

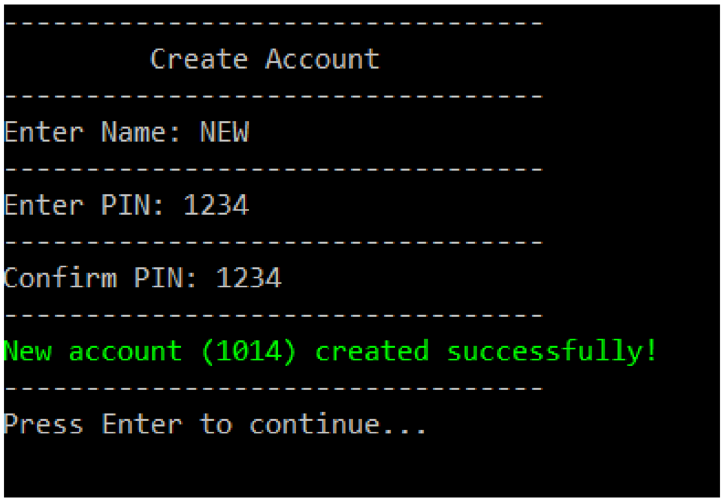


```
-------------------------------------------------------------------
Accounts List:
-------------------------------------------------------------------
 ACCTNO. |   PIN   |     BALANCE       |          NAME          | STATUS
    1001 | 0000    | 3400.00           | Glen                   | Active
    1002 | 1234    | 5000.00           | Angelo                 | Locked
    1003 | 2468    | 11600.00          | Yui                    | Active
    1004 | 030918  | 20500.00          | Happy                  | Active
    1005 | 0427    | 5000.00           | Forodark               | Active
    1006 | 4321    | 4000.00           | Shiki                  | Active
    1007 | 3030    | 5000.00           | Rebecca                | Active
    1008 | 999999  | 7000.00           | Weisz                  | Active
    1009 | 1337    | 12300.00          | Pino                   | Active
    1010 | 8888    | 500.00            | Homura                 | Active
-------------------------------------------------------------------
 Page 1/2    [9] Next Page    [0] Return
-------------------------------------------------------------------
Enter choice: _
```



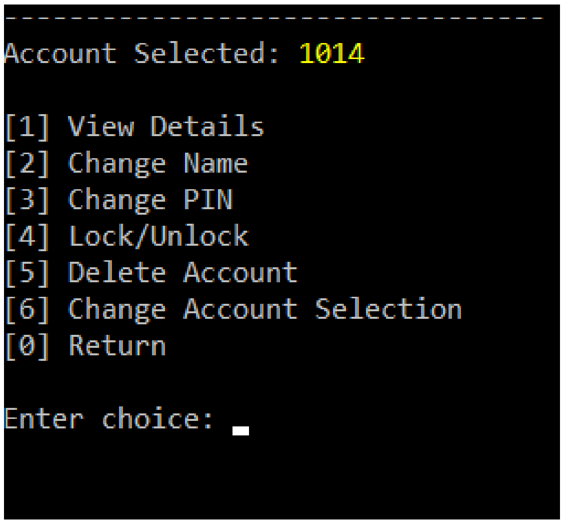
```
-----------------------------------------------------------------
Accounts List:
-----------------------------------------------------------------
 ACCTNO. |   PIN    |      BALANCE      |         NAME         | STATUS
    1011 | 5555     | 1200.00           | Witch                | Active
    1012 | 2222     | 25000.00          | Hermit               | Active
    1013 | 3333     | 4000.00           | Sister               | Active
-----------------------------------------------------------------
 Page 2/2    [8] Previous Page    [0] Return
-----------------------------------------------------------------
Enter choice:
```

[1] List Accounts (Page 1 and 2)



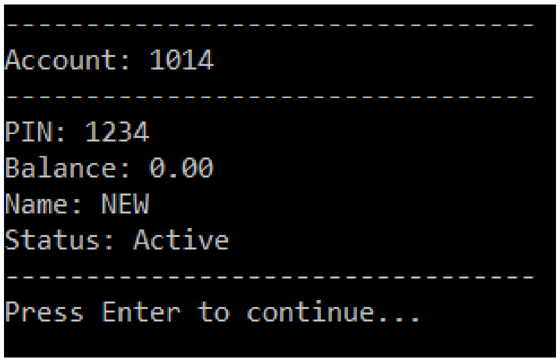
```
---------------------------------
        Create Account
---------------------------------
Enter Name: NEW
---------------------------------
Enter PIN: 1234
---------------------------------
Confirm PIN: 1234
---------------------------------
New account (1014) created successfully!
---------------------------------
Press Enter to continue...
```

[2] Create Account (NEW | 1234)



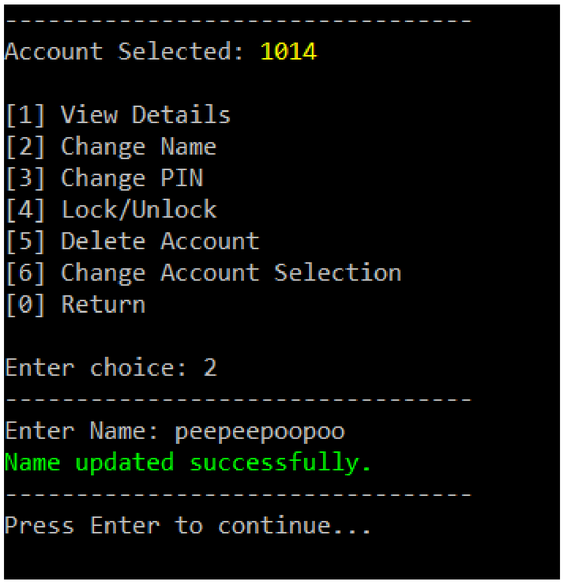
```
---------------------------------
Account Selected: 1014

[1] View Details
[2] Change Name
[3] Change PIN
[4] Lock/Unlock
[5] Delete Account
[6] Change Account Selection
[0] Return

Enter choice:
```

[3] Edit Account (Menu, Account: 1014)

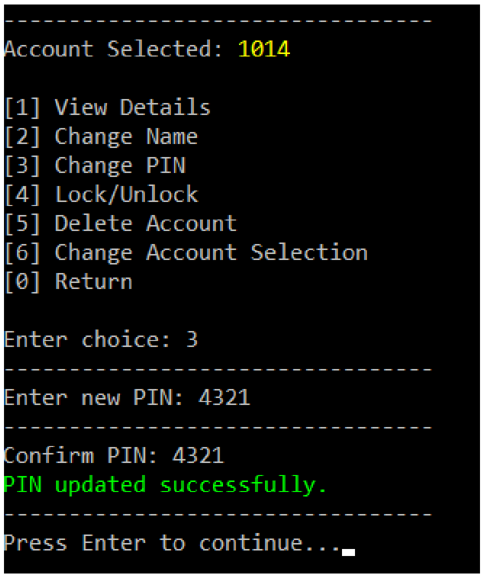


```
---------------------------------
Account: 1014
---------------------------------
PIN: 1234
Balance: 0.00
Name: NEW
Status: Active
---------------------------------
Press Enter to continue...
```

[1] View Details

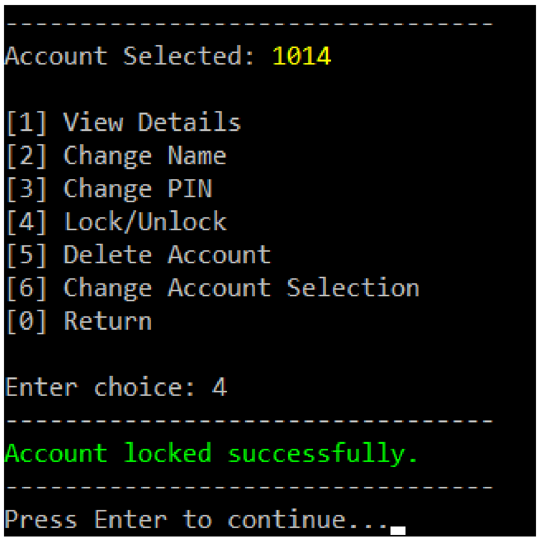


```
---------------------------------
Account Selected: 1014

[1] View Details
[2] Change Name
[3] Change PIN
[4] Lock/Unlock
[5] Delete Account
[6] Change Account Selection
[0] Return

Enter choice: 2
---------------------------------
Enter Name: peepeepoopoo
Name updated successfully.
---------------------------------
Press Enter to continue...
```

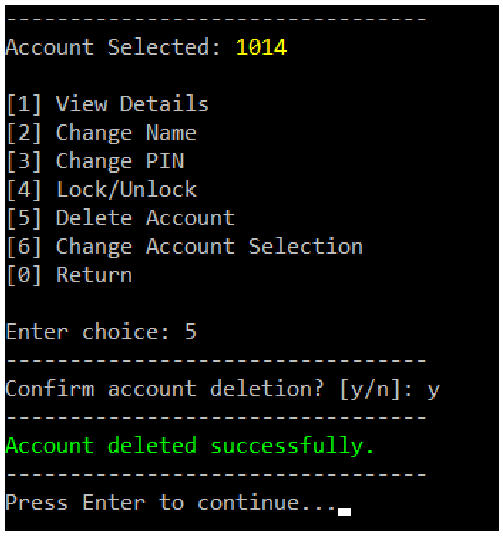


```
---------------------------------
Account Selected: 1014

[1] View Details
[2] Change Name
[3] Change PIN
[4] Lock/Unlock
[5] Delete Account
[6] Change Account Selection
[0] Return

Enter choice: 3
---------------------------------
Enter new PIN: 4321
---------------------------------
Confirm PIN: 4321
PIN updated successfully.
---------------------------------
Press Enter to continue...
```

[2] Change Name, [3] Change PIN

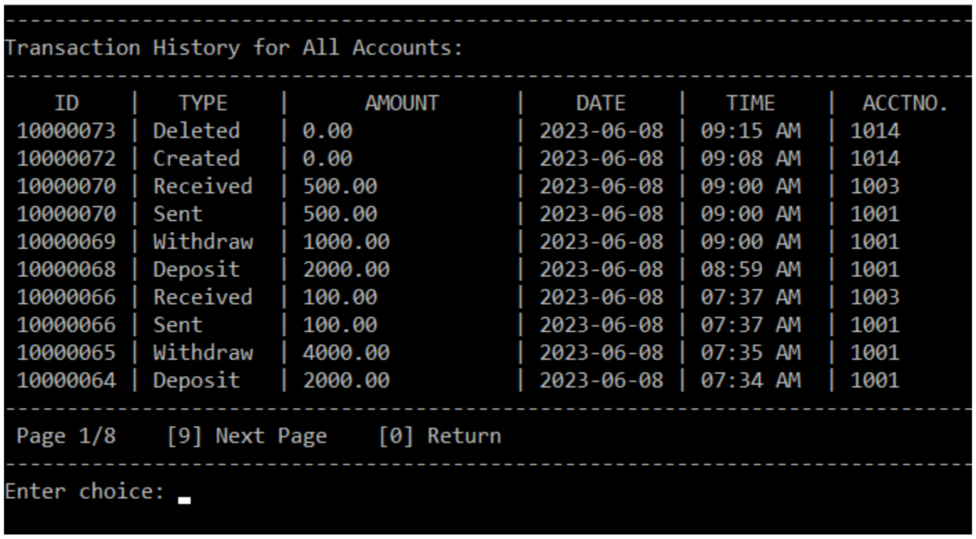


```
----------------------------------
Account Selected: 1014

[1] View Details
[2] Change Name
[3] Change PIN
[4] Lock/Unlock
[5] Delete Account
[6] Change Account Selection
[0] Return

Enter choice: 4
----------------------------------
Account locked successfully.
----------------------------------
Press Enter to continue...
```

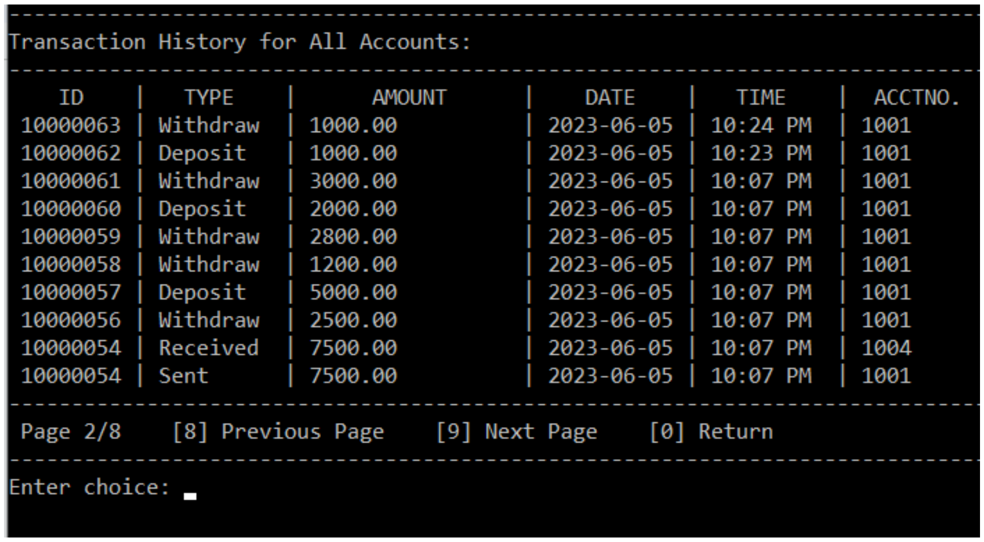


```
----------------------------------
Account Selected: 1014

[1] View Details
[2] Change Name
[3] Change PIN
[4] Lock/Unlock
[5] Delete Account
[6] Change Account Selection
[0] Return

Enter choice: 5
----------------------------------
Confirm account deletion? [y/n]: y
----------------------------------
Account deleted successfully.
----------------------------------
Press Enter to continue...
```

[4] Lock/Unlock, [5] Delete Account



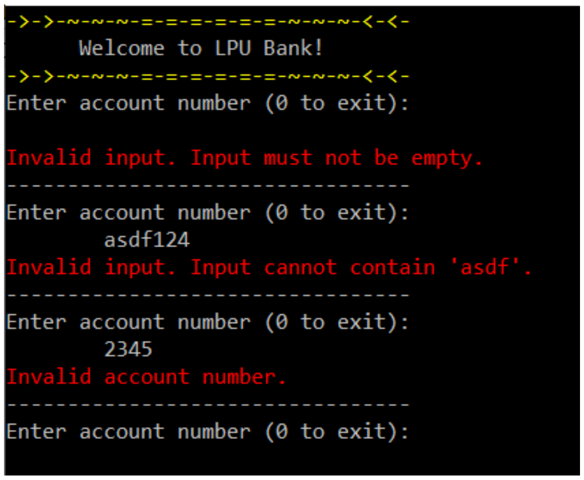
```
------------------------------------------------------------------------------
Transaction History for All Accounts:
------------------------------------------------------------------------------
    ID     |   TYPE    |       AMOUNT       |    DATE    |   TIME    |  ACCTNO.
 10000073 | Deleted   | 0.00               | 2023-06-08 | 09:15 AM  | 1014
 10000072 | Created   | 0.00               | 2023-06-08 | 09:08 AM  | 1014
 10000070 | Received  | 500.00             | 2023-06-08 | 09:00 AM  | 1003
 10000070 | Sent      | 500.00             | 2023-06-08 | 09:00 AM  | 1001
 10000069 | Withdraw  | 1000.00            | 2023-06-08 | 09:00 AM  | 1001
 10000068 | Deposit   | 2000.00            | 2023-06-08 | 08:59 AM  | 1001
 10000066 | Received  | 100.00             | 2023-06-08 | 07:37 AM  | 1003
 10000066 | Sent      | 100.00             | 2023-06-08 | 07:37 AM  | 1001
 10000065 | Withdraw  | 4000.00            | 2023-06-08 | 07:35 AM  | 1001
 10000064 | Deposit   | 2000.00            | 2023-06-08 | 07:34 AM  | 1001
------------------------------------------------------------------------------
 Page 1/8    [9] Next Page    [0] Return
------------------------------------------------------------------------------
Enter choice:
```



```
---------------------------------------------------------------------------------
Transaction History for All Accounts:
---------------------------------------------------------------------------------
    ID     |   TYPE    |      AMOUNT       |    DATE    |    TIME    |  ACCTNO.
 10000063 | Withdraw  | 1000.00           | 2023-06-05 | 10:24 PM  | 1001
 10000062 | Deposit   | 1000.00           | 2023-06-05 | 10:23 PM  | 1001
 10000061 | Withdraw  | 3000.00           | 2023-06-05 | 10:07 PM  | 1001
 10000060 | Deposit   | 2000.00           | 2023-06-05 | 10:07 PM  | 1001
 10000059 | Withdraw  | 2800.00           | 2023-06-05 | 10:07 PM  | 1001
 10000058 | Withdraw  | 1200.00           | 2023-06-05 | 10:07 PM  | 1001
 10000057 | Deposit   | 5000.00           | 2023-06-05 | 10:07 PM  | 1001
 10000056 | Withdraw  | 2500.00           | 2023-06-05 | 10:07 PM  | 1001
 10000054 | Received  | 7500.00           | 2023-06-05 | 10:07 PM  | 1004
 10000054 | Sent      | 7500.00           | 2023-06-05 | 10:07 PM  | 1001
---------------------------------------------------------------------------------
 Page 2/8    [8] Previous Page    [9] Next Page    [0] Return
---------------------------------------------------------------------------------
Enter choice: _
```

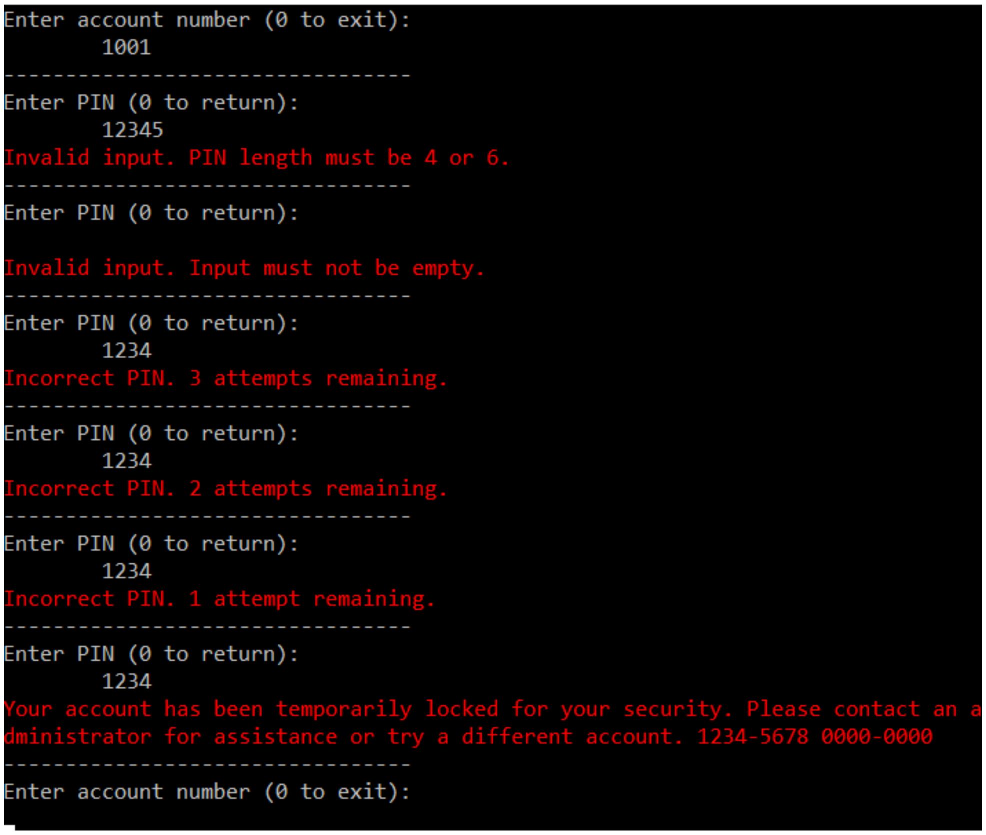
[4] List All Transactions (Page 1 and 2 only)

Sample Output (Invalid Input)



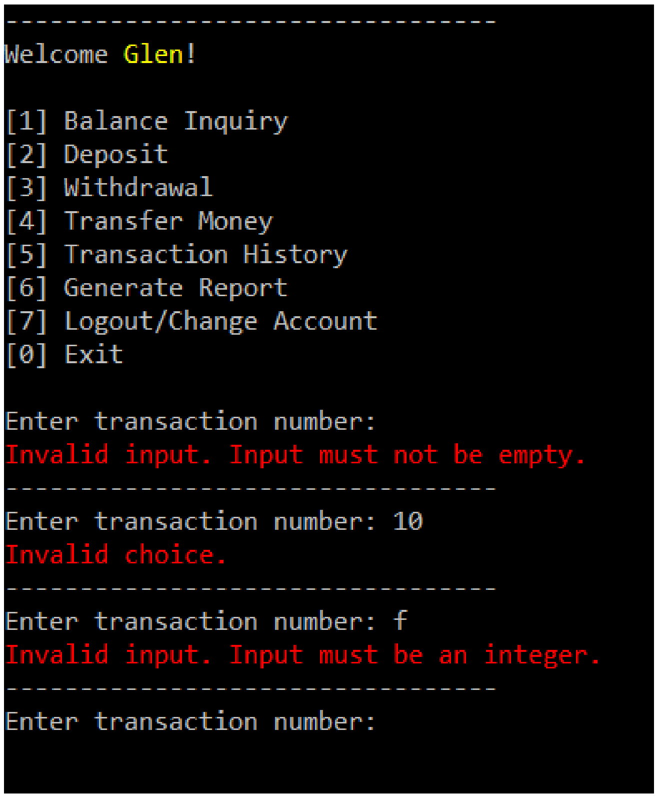
```
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
      Welcome to LPU Bank!
->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-
Enter account number (0 to exit):

Invalid input. Input must not be empty.
---------------------------------
Enter account number (0 to exit):
        asdf124
Invalid input. Input cannot contain 'asdf'.
---------------------------------
Enter account number (0 to exit):
        2345
Invalid account number.
---------------------------------
Enter account number (0 to exit):
```

Login: empty input, non-integer input, invalid account number



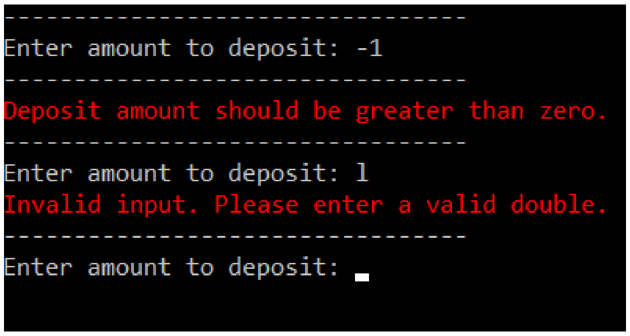
```
Enter account number (0 to exit):
        1001
---------------------------------
Enter PIN (0 to return):
        12345
Invalid input. PIN length must be 4 or 6.
---------------------------------
Enter PIN (0 to return):

Invalid input. Input must not be empty.
---------------------------------
Enter PIN (0 to return):
        1234
Incorrect PIN. 3 attempts remaining.
---------------------------------
Enter PIN (0 to return):
        1234
Incorrect PIN. 2 attempts remaining.
---------------------------------
Enter PIN (0 to return):
        1234
Incorrect PIN. 1 attempt remaining.
---------------------------------
Enter PIN (0 to return):
        1234
Your account has been temporarily locked for your security. Please contact an a
dministrator for assistance or try a different account. 1234-5678 0000-0000
---------------------------------
Enter account number (0 to exit):
```

Login: invalid PIN length, empty PIN, incorrect PIN, locked account



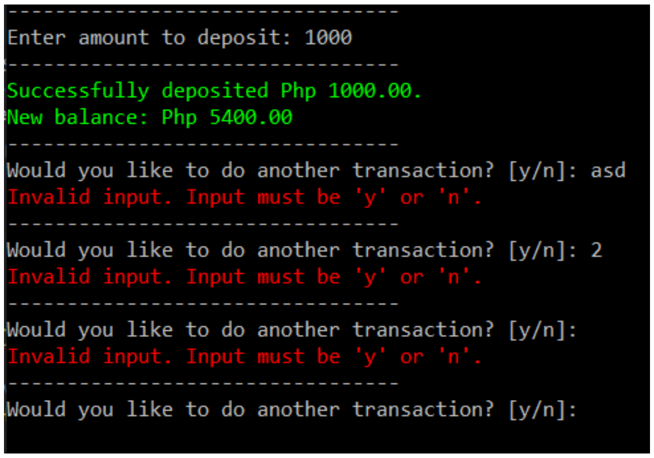
```
--------------------------------
Welcome Glen!

[1] Balance Inquiry
[2] Deposit
[3] Withdrawal
[4] Transfer Money
[5] Transaction History
[6] Generate Report
[7] Logout/Change Account
[0] Exit

Enter transaction number:
Invalid input. Input must not be empty.
--------------------------------
Enter transaction number: 10
Invalid choice.
--------------------------------
Enter transaction number: f
Invalid input. Input must be an integer.
--------------------------------
Enter transaction number:
```

Transaction Menu: empty choice, choice not in menu, non integer choice



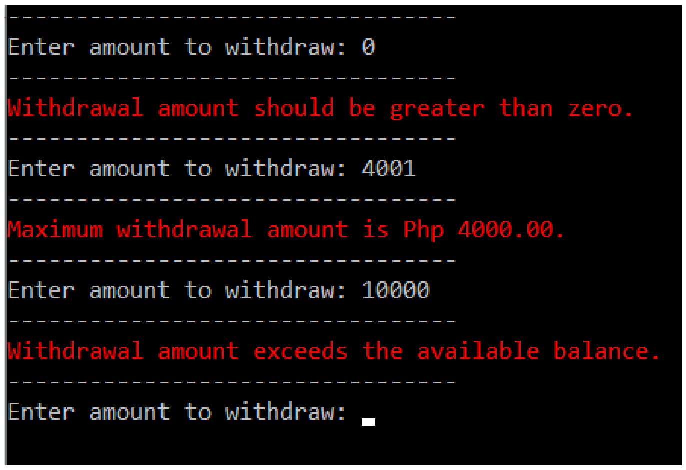
```
--------------------------------
Enter amount to deposit: -1
--------------------------------
Deposit amount should be greater than zero.
--------------------------------
Enter amount to deposit: l
Invalid input. Please enter a valid double.
--------------------------------
Enter amount to deposit: _
```

Deposit: negative amount, non double amount



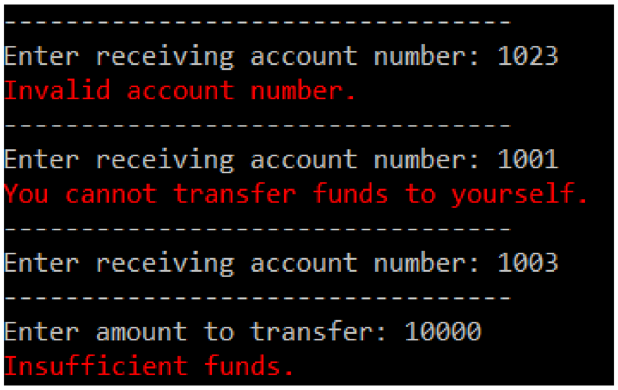
```
-------------------------------
Enter amount to deposit: 1000
-------------------------------
Successfully deposited Php 1000.00.
New balance: Php 5400.00
-------------------------------
Would you like to do another transaction? [y/n]: asd
Invalid input. Input must be 'y' or 'n'.
-------------------------------
Would you like to do another transaction? [y/n]: 2
Invalid input. Input must be 'y' or 'n'.
-------------------------------
Would you like to do another transaction? [y/n]:
Invalid input. Input must be 'y' or 'n'.
-------------------------------
Would you like to do another transaction? [y/n]:
```

New Transaction: invalid inputs



```
-------------------------------
Enter amount to withdraw: 0
-------------------------------
Withdrawal amount should be greater than zero.
-------------------------------
Enter amount to withdraw: 4001
-------------------------------
Maximum withdrawal amount is Php 4000.00.
-------------------------------
Enter amount to withdraw: 10000
-------------------------------
Withdrawal amount exceeds the available balance.
-------------------------------
Enter amount to withdraw:
```

Withdrawal: 0 amount, greater than 4000, insufficient funds



```
-------------------------------
Enter receiving account number: 1023
Invalid account number.
-------------------------------
Enter receiving account number: 1001
You cannot transfer funds to yourself.
-------------------------------
Enter receiving account number: 1003
-------------------------------
Enter amount to transfer: 10000
Insufficient funds.
```

Transfer Money: invalid account, self transfer, insufficient funds

Source Code

```
// ATM V7.2 FINAL VERSION (PROBABLY)
//Admin credentials:
//Account Number: 1000
//Password: admin

//Key features:
//File structure
//Encrypted passwords
//Admin panel
//Create/edit/delete accounts
//Accounts get locked
//Transaction log with page support

//v6 changes
//Fixed deposit function
//Added transfer money feature
//added transaction struct, and display for 5 most recent transactions

//v7 changes
//added view transaction log to admin panel with pages
//changed recent transactions to transaction log with page support
//improved transaction type data type and passing
//added way to exit from entering acct number and pin
//clear command line at certain points to improve readability
//added page support for acc list
//improved variables names for better code readability
//removed newAdminTransaction(), replaced with waitEnter() where its needed
//decluttered and denested as much as i can
//made struct accounts a global variable and removed the parameter passing for accounts
//added view details feature on editAccount()
//added transaction type for deleted account and confirm before account deleting
//separate transaction type for sent and received money, two records will be created one for
each account
//reentered all default account and transaction data
//improved text prompts

//v7.1 changes
//colored text weeeeee
```

```
//v7.2
//added comments
//added global constants

//header files
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "glencrypt.h"

//global constants
#define MAX_NAME 64
#define MAX_WITHDRAWAL 4000
#define MAX_ACCOUNTS 256
#define MAX_TRANSACTIONS 1024
#define PAGE_LENGTH 10
#define MAX_ATTEMPTS 4

//structure for holding account data
struct account {
    char account_number[64];
    char pin[64];
    double balance;
    char name[MAX_NAME];
    int is_locked;
    int deposit_count;
    int withdraw_count;
    double total_deposit;
    double total_withdraw;
};

//structure for holding transaction data
struct transaction {
    char transaction_id[64];
    char account_number[64];
    char type[20];
    double amount;
    char date[20];
    char time[20];
};
```

```
//function prototypes
void login();
void loginAdmin();

void showMenu();
void showAdminMenu();

void balanceInquiry();
void deposit();
void withdrawal();
void transfer();
void generateReport();
void logout();
void exitProgram();

void listAccounts();
void createAccount();
void editAccount();
void viewDetails();
void changeName();
void changePIN();
void lockUnlock();
void deleteAccount();

void newTransaction();
void lockedAccount();
void saveAccounts();
void loadAccounts();
void recordTransaction(int type, double amount);
void listTransactions();
void listAllTransactions();

//declare struct as global
struct transaction transactions[MAX_TRANSACTIONS];
struct account accounts[MAX_ACCOUNTS];

//other global variable definitions
int accounts_size, transactions_size;
int selection = -1; int transfer_to = -1;

int main() {
        printColor(YELLOW,"->->-~-~-~-=-=-=-=-=-=-=-~-~-~-<-<-\n");
    printf("%27s", "Welcome to LPU Bank!\n");
    printColor(YELLOW,"->->-~-~-~-=-=-=-=-=-=-=-~-~-~-<-<-\n");

        //set account size and transaction size to 0, this will later be modified whenever the struct
for accounts or transactions is loaded
    accounts_size = 0; transactions_size = 0;
    loadAccounts(); // load struct accounts from accounts.dat

    login();

    return 0;
}

void login() { //loop for checking account number
    int attempts = MAX_ATTEMPTS; //set attempts to max attempts (4)
    int i; //declare i, a temporary variable for checking if the account is valid
    while(1) {
        char account_number[64];
        int valid = 0; //temporary variable that is set to 1 if the account is valid
        inputCustom("Enter account number (0 to exit):\n\t", &account_number, "1234567890");

        if (!strcmp("0", account_number)) { //exit if 0 is entered
            exitProgram();
                }
        if (!strcmp("1000", account_number)) { //switch to admin login if 1000 is entered
                        loginAdmin();
                }

        for (i = 0; i < accounts_size; i++) { //loop for checking account validity
            if (!strcmp(accounts[i].account_number, account_number)) {
                                valid = 1;
                                break;
            }
            }
            if (valid == 0) { //if not valid, then report error, then repeat loop
                    printColor(RED,"Invalid account number.\n");
                        printLine(0);
                continue;
                }
        if (accounts[i].is_locked == 1) { //if valid but locked, then give locked prompt, and function
gets repeated
            lockedAccount();
        }
        printLine(0);
        break; //if none of the error checks are met, then breaks the while loop, to move on to next
loop
            }

    while(attempts > 0) { //loop for checking pin number
        char pin[64];
        inputCustom("Enter PIN (0 to return):\n\t", &pin, "1234567890");

        if (!strcmp("0", pin)) { //if 0 is entered, return to account number login
            printLine(0);
            login();
                }
                if (strlen(pin) != 6 && strlen(pin) != 4) { //if invaild pin length, restart loop
            printColor(RED,"Invalid input. PIN length must be 4 or 6.\n");
            printLine(0);
                        continue;
        }
                if (!strcmp(decrypt(accounts[i].pin), pin)) { //if match is found, then set the
selection variable to the value of i, so that account is used for the rest of the program
                    selection = i;
                    showMenu();
        }
        attempts -= 1; // every time pin is incorrect attempts gets deducted by 1
        if (attempts > 1) { //if its still greater than 1, prompt with attempts remaning
            printColor(RED,"Incorrect PIN. %d attempts remaining.\n", attempts);
            printLine(0);
        }
        else if (attempts == 1) {
            printColor(RED,"Incorrect PIN. %d attempt remaining.\n", attempts);
            printLine(0);
        }
        else { //if no attempts left, lock the account
            accounts[i].is_locked = 1;
            lockedAccount();
        }
    }
```

```
}

void loginAdmin() { //function to handle admin password login
   printLine(0);
         while(1) {
      char pass[64];
      char admin_pass[] = "7#F5m"; //admin
      inputCustom("Enter Password (0 to return):\n\t", &pass, ALPHANUM);

      if (!strcmp("0", pass)) {
         printLine(0);
         login();
                  }

      if (!strcmp(decrypt(admin_pass), pass)) {
         showAdminMenu();
         return;
      }
      else {
         printColor(RED,"Incorrect Password.\n");
      }
      printLine(0);
         }
}

void showMenu() { //function to show the menu, and handle the selection
   int choice;
   system("CLS");
   printLine(0);
   printf("Welcome ");
         printColor(YELLOW,"%s", accounts[selection].name);
         printf("!\n\n");
   printf("[1] Balance Inquiry\n");
   printf("[2] Deposit\n");
   printf("[3] Withdrawal\n");
   printf("[4] Transfer Money\n");
   printf("[5] Transaction History\n");
   printf("[6] Generate Report\n");
   printf("[7] Logout/Change Account\n");
   printf("[0] Exit\n\n");
   while (1) {
        input(Int, "Enter transaction number: ", &choice);
        switch (choice) {
            case 1:
                balanceInquiry();
                break;
            case 2:
                deposit();
                break;
            case 3:
                withdrawal();
                break;
            case 4:
                        transfer();
                break;
            case 5:
                        listTransactions();
                break;
            case 6:
                generateReport();
                break;
            case 7:
                logout();
                break;
            case 0:
                exitProgram();
            default:
                printColor(RED,"Invalid choice.\n");
                printLine(0);
                continue;
        }
        break;
    }
}

void showAdminMenu() { //menu for admin
            saveAccounts();
    int choice;
    system("CLS");
    printLine(0);
    printf("Welcome ");
            printColor(YELLOW,"Admin");
            printf("!\n\n");
    printf("[1] List Accounts\n");
    printf("[2] Create Account\n");
    printf("[3] Edit Account\n");
    printf("[4] List All Transactions\n");
    printf("[0] Logout\n\n");
    while (1) {
        input(Int, "Enter transaction number: ", &choice);
        switch (choice) {
            case 1:
                listAccounts();
                break;
            case 2:
                createAccount();
                break;
            case 3:
                        editAccount();
                break;
            case 4:
                listAllTransactions();
                break;
            case 0:
                logout();
                break;
            default:
                printColor(RED,"Invalid choice.\n");
                printLine(0);
                continue;
        }
        break;
    }
}

void balanceInquiry() { //prints the accuont balance
            system("CLS");
            printLine(0);
    printf("Account balance: Php %.2lf\n", accounts[selection].balance);
    printLine(0);
    newTransaction();
}
```

```
void deposit() { //asks the user for an amount, then adds it to the account balance
          system("CLS");
          printLine(0);
          while(1) {
             double amount;
             input(Double, "Enter amount to deposit: ", &amount);
             printLine(0);
             if (amount <= 0) {
                 printColor(RED,"Deposit amount should be greater than zero.\n");
                 printLine(0);
                 continue;
             }
       accounts[selection].balance += amount;
       printColor(GREEN,"Successfully deposited Php %.2lf.\nNew balance: Php %.2lf\n",
amount, accounts[selection].balance);
       accounts[selection].deposit_count += 1;
       accounts[selection].total_deposit += amount;
       recordTransaction(2, amount);
       printLine(0);
          newTransaction();
          }
}

void withdrawal() { //asks the user for an amount, then deducts it to the account balance
          system("CLS");
          printLine(0);
    while (1) {
       double amount;
       input(Double, "Enter amount to withdraw: ", &amount);
       printLine(0);
       if (amount > accounts[selection].balance) {
           printColor(RED,"Withdrawal amount exceeds the available balance.\n");
           printLine(0);
           continue;
       }
       if (amount <= 0) {
           printColor(RED,"Withdrawal amount should be greater than zero.\n");
           printLine(0);
           continue;
       }
       if (amount > MAX_WITHDRAWAL) {
           printColor(RED,"Maximum withdrawal amount is Php 4000.00.\n");
           printLine(0);
           continue;
       }
       accounts[selection].balance -= amount;
       printColor(GREEN,"Withdrawal successful.\nNew balance: Php %.2lf\n",
accounts[selection].balance);
       accounts[selection].withdraw_count += 1;
       accounts[selection].total_withdraw += amount;
       recordTransaction(3, amount);
       printLine(0);
       newTransaction();
    }
}

void transfer() { //asks the user for a vaild account number and an amount, then transfer that to
the given account number
           system("CLS");
           printLine(0);
    while(1) {
                    char account_number[64];
             inputCustom("Enter receiving account number: ", &account_number, "1234567890");
             if (!strcmp(accounts[selection].account_number, account_number)) {
                    printColor(RED,"You cannot transfer funds to yourself.\n");
                    printLine(0);
                    continue;
                    }
                    for (int i = 0; i < accounts_size; i++) {
           if (!strcmp(accounts[i].account_number, account_number)) {
             transfer_to = i;
             printLine(0);
           }
                    }
                    if (transfer_to == -1) {
                            printColor(RED,"Invalid account number.\n");
                            printLine(0);
                            continue;
                    }
                    break;
           }
        while(1) {
            double amount;
            input(Double, "Enter amount to transfer: ", &amount);

            if (amount <= 0) {
                printColor(RED,"Transfer amount should be greater than zero.\n");
                printLine(0);
                continue;
            }
            if (amount > accounts[selection].balance) {
                    printColor(RED,"Insufficient funds.\n");
                    printLine(0);
                    continue;
                    }
        printLine(0);
    accounts[selection].balance -= amount;
    accounts[transfer_to].balance += amount;
    printColor(GREEN,"Successfully transferred Php %.2lf to %s (%s).\nNew balance: Php
%.2lf\n",
                amount, accounts[transfer_to].name, accounts[transfer_to].account_number,
accounts[selection].balance);
        recordTransaction(4, amount);
    transfer_to = -1;
                printLine(0);
        newTransaction();
        }
}

void generateReport() { //reports the count for deposit and withdrawal, and the total amount
        system("CLS");
        printLine(0);
  printf("Deposit Count: %d\n", accounts[selection].deposit_count);
  printf("Withdrawal Count: %d\n", accounts[selection].withdraw_count);
  printf("Total Amount Deposited: %.2lf\n", accounts[selection].total_deposit);
  printf("Total Amount Withdrawn: %.2lf\n", accounts[selection].total_withdraw);
  printLine(0);
  newTransaction();
}

void logout() { //logs out, calls the login function again
        system("CLS");
        printLine(0);
    selection = -1;
    printf("Logging out...\n");
    printLine(0);
    login();
}

void exitProgram() { //saves all changes, and exits program
        saveAccounts();
        printColor(YELLOW,"->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-\n");
    printf("Thank you for transacting with LPU Bank. <3\n");
    printColor(YELLOW,"->->-~-~-~-=-=-=-=-=-=-~-~-~-<-<-\n");
    exit(0);
}

void listAccounts() { //lists all the accounts in the system with a progressively generated table
format and page support

        int i; int j = 0;
        int page = 0; int max_page = (accounts_size-1)/PAGE_LENGTH;
        int item_count = 0;

        while(1) {
                system("CLS");
                printLine(2);
            printf("Accounts List:\n");
                printLine(2);
            printf(" ACCTNO. |   PIN   |     BALANCE     |        NAME        | STATUS\n");
            for(i=item_count; j<PAGE_LENGTH && i<accounts_size ; i++,j++) {
                char status[8];
                if(accounts[i].is_locked == 0) {
                        strcpy(status, "Active");
                        }
                        else if(accounts[i].is_locked == 1) {
                        strcpy(status, "Locked");
                        }
                printf(" %7s | %-6s  | %-15.2lf | %-18s | %6s\n", accounts[i].account_number,
decrypt(accounts[i].pin), accounts[i].balance, accounts[i].name, status);
                }
                j = 0;
            printLine(2);
                printf(" Page %d/%d    ", page+1, max_page+1);
                if (page > 0) {
                        printf("[8] Previous Page    ");
                }
                if (page != max_page) {
                        printf("[9] Next Page    ");
                }
                printf("[0] Return\n");
                printLine(2);

                int choice;
                while (1) {
            input(Int, "Enter choice: ", &choice);
            switch (choice) {
                case 8:
                        if (page > 0) {
                                                page--;
                                                item_count = page*PAGE_LENGTH;
                                                break;
                                        }
                                        else {
                            printColor(RED,"Invalid choice.\n");
                            printLine(0);
                            continue;
                                        }

                case 9:
                        if (page != max_page) {
                                page++;
                                item_count = page*PAGE_LENGTH;
                                                break;
                                        }
                                        else {
                            printColor(RED,"Invalid choice.\n");
                            printLine(0);
                            continue;
                                        }
                case 0:
                                        printLine(2);
                                    showAdminMenu();
                    break;
```

```
                    default:
                        printColor(RED,"Invalid choice.\n");
                        printLine(0);
                        continue;
                }
                break;
            }
        }
}

void createAccount() { //asks the user for a name and pin, the rest of the struct account
members are generated automatically
            system("CLS");
            printLine(0);
    if (accounts_size >= MAX_ACCOUNTS) {
        printColor(RED,"Maximum number of accounts reached. Cannot create a new account.\n");
        waitEnter();
        showAdminMenu();
    }

    printf("%24s", "Create Account\n");
    printLine(0);

    sprintf(accounts[accounts_size].account_number, "%d", 1000 + accounts_size + 1);

    inputCustom("Enter Name: ", accounts[accounts_size].name, ALPHANUM);
    printLine(0);
    while (1) {
            char pin[64];
        inputCustom("Enter PIN: ", &pin, "0123456789");
        if (strlen(pin) != 6 && strlen(pin) != 4) {
            printColor(RED,"Invalid input. PIN length must be 4 or 6.\n");
            printLine(0);
            continue;
        }
        char confirm_pin[64];
        printLine(0);
        inputCustom("Confirm PIN: ", confirm_pin, "0123456789");
        if (strcmp(pin, confirm_pin) == 0) {
            strcpy(accounts[accounts_size].pin, encrypt(pin));
            printLine(0);
            break;
        } else {
            printColor(RED,"PINs do not match. Please try again.\n");
            printLine(0);
        }
    }
    accounts[accounts_size].balance = 0.0;

    accounts[accounts_size].is_locked = 0;
    accounts[accounts_size].deposit_count = 0;
    accounts[accounts_size].withdraw_count = 0;
    accounts[accounts_size].total_deposit = 0.0;
    accounts[accounts_size].total_withdraw = 0.0;

            selection = accounts_size;
            recordTransaction(0, 0.0);
            selection = -1;
    printColor(GREEN,"New account (%s) created successfully!\n",
accounts[accounts_size].account_number);
    printLine(0);
    accounts_size += 1;
    waitEnter();
    showAdminMenu();
}

void editAccount() { //asks user for valid account number, then shows menu with different
options
            system("CLS");
            printLine(0);
    if (accounts_size == 0) {
        printColor(RED,"No accounts found.\n");
        printLine(0);
        waitEnter();
        showAdminMenu();
    }
            char account_number[64];
            while(1) {
                inputCustom("Enter account number: ", &account_number, "1234567890");

                    for (int i = 0; i < accounts_size; i++) {
                if (!strcmp(accounts[i].account_number, account_number)) {
                            selection = i;
                    }
                    }
                    if (selection == -1) {
                            printColor(RED,"Invalid account number.\n");
                            printLine(0);
                            continue;
                    }
                    break;
            }

            while(1) { //edit menu and selection
                    int choice;
                    system("CLS");
                    printLine(0);
                    printf("Account Selected: ");
                    printColor(YELLOW, "%s\n\n", account_number);
                    printf("[1] View Details\n");
                    printf("[2] Change Name\n");
        printf("[3] Change PIN\n");
        printf("[4] Lock/Unlock\n");
            printf("[5] Delete Account\n");
            printf("[6] Change Account Selection\n");
            printf("[0] Return\n\n");
                while (1) {
                input(Int, "Enter choice: ", &choice);
                     switch (choice) {
                            case 1:
                                    viewDetails();
                                    break;
                         case 2:
                            changeName();
                            break;
                         case 3:
                                            changePIN();
                                            break;
                         case 4:
                                            lockUnlock();
                            break;
                case 5:
                                            deleteAccount();
                    break;
                case 6:
                                            selection = -1;
                                            editAccount();
                    break;
                case 0:
                      printLine(0);
                      selection = -1;
                      showAdminMenu();
                      break;
                            default:
                               printColor(RED,"Invalid choice. Please try again.\n");
                               printLine(0);
                                            continue;
                         }
                         printLine(0);
                         waitEnter();
                         break;
                }
            }
}

void viewDetails() { //shows account info
        char status[8];
        if(accounts[selection].is_locked == 0) {
                strcpy(status, "Active");
        }
        else if(accounts[selection].is_locked == 1) {
                strcpy(status, "Locked");
        }

        system("CLS");
        printLine(0);
        printf("Account: %s\n", accounts[selection].account_number);
        printLine(0);
        printf("PIN: %s\n", decrypt(accounts[selection].pin));
        printf("Balance: %.2lf\n", accounts[selection].balance);
        printf("Name: %s\n", accounts[selection].name);
        printf("Status: %s\n", status);
}
```

```
void changeName() { //asks user for name, and updates the account name
   char new_name[MAX_NAME];
         printLine(0);
   inputCustom("Enter Name: ", new_name, ALPHANUM);
   strcpy(accounts[selection].name, new_name);
   printColor(GREEN,"Name updated successfully.\n");
}

void changePIN() { //asks user for a valid pin, and updates the account pin
   while (1) {
         printLine(0);
      char new_pin[64];
      inputCustom("Enter new PIN: ", new_pin, "0123456789");
      printLine(0);
      if (strlen(new_pin) != 6 && strlen(new_pin) != 4) {
         printColor(RED,"Invalid input. PIN length must be 4 or 6.\n");
      }
      else {
         char confirmPin[64];
         inputCustom("Confirm PIN: ", confirmPin, "0123456789");
         if (!strcmp(new_pin, confirmPin)) {
            strcpy(accounts[selection].pin, encrypt(new_pin));
            printColor(GREEN,"PIN updated successfully.\n");
            break;
         }
                           else {
            printColor(RED,"PINs do not match. Please try again.\n");
            printLine(0);
         }
                  }

   }
}

void lockUnlock() { //locks or unlocks the account
         printLine(0);
   accounts[selection].is_locked = !accounts[selection].is_locked;
   if (accounts[selection].is_locked) {
      printColor(GREEN,"Account locked successfully.\n");
   }
         else {
```

```
        printColor(GREEN,"Account unlocked successfully.\n");
    }
}

void deleteAccount() { //deletes the account, and shifts the position of each struct in the array
    int choice;
    printLine(0);
    input(Bool, "Confirm account deletion? [y/n]: ", &choice);
        if (choice == 0) {
                printLine(0);
                printColor(RED,"Account deletion cancelled.\n");
        return;
    }
    for (int j = selection; j < accounts_size - 1; j++) {
        accounts[j] = accounts[j + 1];
    }
    accounts_size += -1;
    recordTransaction(1, 0.0);
    printLine(0);
    printColor(GREEN,"Account deleted successfully.\n");
    selection = -1;
    printLine(0);
    waitEnter();
    showAdminMenu();
}

void newTransaction() { //asks the user if they want to do another transaction, return to menu if
yes, exit if no
        saveAccounts();
    int choice;
    input(Bool, "Would you like to do another transaction? [y/n]: ", &choice);
    if (choice == 1) {
        printLine(0);
        showMenu();
        return;
    }
    else if (choice == 0) {
        exitProgram();
        return;
    }
}
```

```
void lockedAccount() { //showws the locked account prompt, and returns to login
   printColor(RED,"Your account has been temporarily locked for your security. Please contact
an administrator for assistance or try a different account. 1234-5678 0000-0000\n");
   printLine(0);
   login();
}

void saveAccounts() { //saves the account by writing to the accounts.dat file
   FILE* file = fopen("accounts.dat", "wb");
   if (file == NULL) {
      printColor(RED,"Error: Unable to save accounts.\n");
      printLine(0);
                  return;
   }
   fwrite(accounts, sizeof(struct account), accounts_size, file);
   fclose(file);
}

void loadAccounts() { //loads the account by reading the accounts.dat file
   FILE* file = fopen("accounts.dat", "rb");
   if (file == NULL) {
      printColor(RED,"Error: Unable to load accounts.\n");
      printLine(0);
      return;
   }
   accounts_size = fread(accounts, sizeof(struct account), MAX_ACCOUNTS, file);
   fclose(file);
}

void recordTransaction(int type, double amount) { //records a transaction, and assigns an id to it
   struct transaction new_transaction;

         FILE *file = fopen("transactions.dat", "ab");
         if (file == NULL) {
      printColor(RED,"Failed to open the file.\n");
      return;
   }

   fseek(file, 0, SEEK_END);
   long file_size = ftell(file);
long num_transactions = file_size / sizeof(struct transaction);

sprintf(new_transaction.transaction_id, "%d", 10000000 + num_transactions + 1);
strcpy(new_transaction.account_number, accounts[selection].account_number);
new_transaction.amount = amount;

time_t currentTime = time(NULL);
struct tm *localTime = localtime(&currentTime);
strftime(new_transaction.date, sizeof(new_transaction.date), "%Y-%m-%d", localTime);
strftime(new_transaction.time, sizeof(new_transaction.time), "%I:%M %p", localTime);

    if (type == 0) {
            strcpy(new_transaction.type, "Created");
    }
    else if (type == 1) {
            strcpy(new_transaction.type, "Deleted");
    }
    else if (type == 2) {
            strcpy(new_transaction.type, "Deposit");
    }
    else if (type == 3) {
            strcpy(new_transaction.type, "Withdraw");
    }
    else if (type == 4){
            strcpy(new_transaction.type, "Sent");
    }
    else {
            strcpy(new_transaction.type, "Unknown");
    }


fwrite(&new_transaction, sizeof(struct transaction), 1, file);

if (type == 4) {
    strcpy(new_transaction.type, "Received");
    strcpy(new_transaction.account_number, accounts[transfer_to].account_number);
    fwrite(&new_transaction, sizeof(struct transaction), 1, file);
    }

fclose(file);
```

```
   transactions_size += 1;
}

void listTransactions() { //lists all transactions associated with a specific using a progressively
generated table format with page support
   struct transaction transactions[MAX_TRANSACTIONS];
   struct transaction account_transactions[MAX_TRANSACTIONS];

         FILE *file = fopen("transactions.dat", "rb");
         if (file == NULL) {
      printColor(RED,"Failed to open the file.\n");
      return;
   }
   transactions_size = fread(transactions, sizeof(struct transaction), MAX_TRANSACTIONS,
file);

         int account_transactions_size = 0;
         for(int k = 0; k < transactions_size; k++) {
            if (!strcmp(accounts[selection].account_number, transactions[k].account_number)) {
                     account_transactions[account_transactions_size] = transactions[k];
                     account_transactions_size++;
                  }
         }


         int i; int j = 0;
         int page = 0; int max_page = (account_transactions_size-1)/PAGE_LENGTH;
         int item_count = account_transactions_size-1-(page*PAGE_LENGTH);

         while(1) {
                  system("CLS");
                  printLine(2);
            printf("Transaction History for Account %s:\n", accounts[selection].account_number);
                  printLine(2);
            printf("    ID    |  TYPE    |      AMOUNT       |   DATE    |  TIME    \n");
                  for(i=item_count; j<PAGE_LENGTH && i>=0 ; i--, j++) {
                  printf(" %8s | %-10s| %-16.2lf | %-10s | %-11s\n",
account_transactions[i].transaction_id, account_transactions[i].type,
account_transactions[i].amount, account_transactions[i].date, account_transactions[i].time);
                  }
                  j = 0;
```

```
        printLine(2);
            printf(" Page %d/%d    ", page+1, max_page+1);
            if (page > 0) {
                printf("[8] Previous Page    ");
            }
            if (page != max_page) {
                printf("[9] Next Page    ");
            }
            printf("[0] Return\n");
        printLine(2);

            int choice;
            while (1) {
          input(Int, "Enter choice: ", &choice);
          switch (choice) {
            case 8:
                if (page > 0) {
                                page--;
                                item_count =
account_transactions_size-1-(page*PAGE_LENGTH);
                                break;
                            }
                            else {
                      printColor(RED,"Invalid choice.\n");
                      printLine(0);
                      continue;
                            }

            case 9:
                if (page != max_page) {
                    page++;
                    item_count =
account_transactions_size-1-(page*PAGE_LENGTH);
                                break;
                            }
                            else {
                      printColor(RED,"Invalid choice.\n");
                      printLine(0);
                      continue;
                            }
            case 0:
```

```
                                                fclose(file);
                                                printLine(0);
                                        newTransaction();
                          break;
                        default:
                          printColor(RED,"Invalid choice.\n");
                          printLine(0);
                          continue;
                    }
                    break;
                }
            }
}

void listAllTransactions() { //lists all transactions using a progressively generated table format
with page support
    struct transaction transactions[MAX_TRANSACTIONS];

            FILE *file = fopen("transactions.dat", "rb");
            if (file == NULL) {
        printColor(RED,"Failed to open the file.\n");
        return;
    }
    transactions_size = fread(transactions, sizeof(struct transaction), MAX_TRANSACTIONS,
file);

            int i; int j = 0;
            int page = 0; int max_page = (transactions_size-1)/PAGE_LENGTH;
            int item_count = transactions_size-1-(page*PAGE_LENGTH);

            while(1) {
                        system("CLS");
                        printf("-------------------------------------------------------------------------\n");
                printf("Transaction History for All Accounts:\n");
                        printf("-------------------------------------------------------------------------\n");
                printf("    ID    |  TYPE   |      AMOUNT      |   DATE    |   TIME    |  ACCTNO.  \n");
                for(i=item_count; j<PAGE_LENGTH && i>=0 ; i--, j++) {
                        printf(" %8s | %-10s| %-16.2lf | %-10s | %-9s | %-7s \n",
transactions[i].transaction_id, transactions[i].type, transactions[i].amount, transactions[i].date,
transactions[i].time, transactions[i].account_number);
                        }
                        j = 0;
                        printf("------------------------------------------------------------------------------\n");
                        printf(" Page %d/%d    ", page+1, max_page+1);
                        if (page > 0) {
                            printf("[8] Previous Page    ");
                        }
                        if (page != max_page) {
                            printf("[9] Next Page    ");
                        }
                        printf("[0] Return\n");
                        printf("------------------------------------------------------------------------------\n");

                        int choice;
                        while (1) {
                    input(Int, "Enter choice: ", &choice);
                    switch (choice) {
                        case 8:
                            if (page > 0) {
                                        page--;
                                        item_count =
transactions_size-1-(page*PAGE_LENGTH);
                                        break;
                                    }
                                    else {
                               printColor(RED,"Invalid choice.\n");
                               printLine(0);
                               continue;
                                    }

                        case 9:
                            if (page != max_page) {
                                page++;
                                item_count = transactions_size-1-(page*PAGE_LENGTH);
                                        break;
                                    }
                                    else {
                               printColor(RED,"Invalid choice.\n");
                               printLine(0);
                               continue;
                                    }
                        case 0:
```

```
                                        fclose(file);
                                        printLine(0);
                                    showAdminMenu();
                        break;
                      default:
                        printColor(RED,"Invalid choice.\n");
                        printLine(0);
                        continue;
                    }
                    break;
                  }
                }
}
```