

**CISC 327 Assignment 6**

**Scott Wallace 10051890**

**Brad Guner 10059112**

**Daily Script**

#!/bin/bash

cd DailySessions #Enter directory with inputs

for i in \*.txt #Go through every txt file, each

do #its own session

echo "running $i" #

FILE="$i" #

cat $FILE | while read line #This is how we automate inputs

do #on front end.

echo $line #

done | python ../breakingbank.py #

cat ../SummaryFiles/\* >> ../mergedtransactions.txt #merge to merged transactions

done #

cd .. #

python breakingbank-backend.py #Run back-end at end of day

**Weekly Script**

#!/bin/bash

echo "=== Day 1 ==="

cp WeeklySessions/Day1/\* DailySessions #Where inputs are stored for Day1

./daily #Run daily script

cd DailySessions #Clean out DailySessions Directory

rm \* #

cd .. #

cp SummaryFiles/\* OldSummaryFiles #Move summary files created into storage

cd SummaryFiles #Clean out SummaryFiles Directory

rm \* #

cd .. #

cat mergedtransactions.txt > Day1Merged.txt #Save merged trans for Day 1

cat masteraccounts.txt > Day1Master.txt #Save master accounts for Day 1

echo -n "" > mergedtransactions.txt #Empty our mergedtransactions.txt for next day

echo "=== Day 2 ==="

cp WeeklySessions/Day2/\* DailySessions

./daily

cd DailySessions

rm \*

cd ..

cp SummaryFiles/\* OldSummaryFiles

cd SummaryFiles

rm \*

cd ..

cat mergedtransactions.txt > Day2Merged.txt

cat masteraccounts.txt > Day2Master.txt

echo -n "" > mergedtransactions.txt

echo "=== Day 3 ==="

cp WeeklySessions/Day3/\* DailySessions

./daily

cd DailySessions

rm \*

cd ..

cp SummaryFiles/\* OldSummaryFiles

cd SummaryFiles

rm \*

cd ..

cat mergedtransactions.txt > Day3Merged.txt

cat masteraccounts.txt > Day3Master.txt

echo -n "" > mergedtransactions.txt

echo "=== Day 4 ==="

cp WeeklySessions/Day4/\* DailySessions

./daily

cd DailySessions

rm \*

cd ..

cp SummaryFiles/\* OldSummaryFiles

cd SummaryFiles

rm \*

cd ..

cat mergedtransactions.txt > Day4Merged.txt

cat masteraccounts.txt > Day4Master.txt

echo -n "" > mergedtransactions.txt

echo "=== Day 5 ==="

cp WeeklySessions/Day5/\* DailySessions

./daily

cd DailySessions

rm \*

cd ..

cp SummaryFiles/\* OldSummaryFiles

cd SummaryFiles

rm \*

cd ..

cat mergedtransactions.txt > Day5Merged.txt

cat masteraccounts.txt > Day5Master.txt

echo -n "" > mergedtransactions.txt

**Set of Transaction inputs for Day 3**

|  |  |  |
| --- | --- | --- |
| session3-1.txt (Agent) | session3-2.txt (Retail) | session3-3.txt (Retail |
| withdraw, 000005, 100.00 | deposit, 000001, 150.00 | deposit, 000001, 150.00 |
| delete, 000005, M Ehrmantraut | deposit, 000002, 150.00 | deposit, 000002, 150.00 |
| withdraw, 000006, 100.00 | withdraw, 000003, 20.00 | deposit, 000003, 200.00 |
| delete, 000006, Lydia Rodarte | withdraw, 000004, 10.00 | deposit, 000004, 300.00 |

**Merged Transactions for Day 3**

02\_000005\_BBBBBB\_00010000\_NNNNNNNNNNNNNNN

02\_000006\_BBBBBB\_00010000\_NNNNNNNNNNNNNNN

05\_000005\_BBBBBB\_00000000\_M Ehrmantraut

05\_000006\_BBBBBB\_00000000\_Lydia Rodarte

01\_000001\_BBBBBB\_00015000\_NNNNNNNNNNNNNNN

01\_000002\_BBBBBB\_00015000\_NNNNNNNNNNNNNNN

02\_000003\_BBBBBB\_00002000\_NNNNNNNNNNNNNNN

02\_000004\_BBBBBB\_00001000\_NNNNNNNNNNNNNNN

01\_000001\_BBBBBB\_00015000\_NNNNNNNNNNNNNNN

01\_000002\_BBBBBB\_00015000\_NNNNNNNNNNNNNNN

01\_000003\_BBBBBB\_00020000\_NNNNNNNNNNNNNNN

01\_000004\_BBBBBB\_00030000\_NNNNNNNNNNNNNNN

**Master Accounts Day 1**

000001\_00000000\_000Steven Gomez

000002\_00000000\_00000000Tortuga

000003\_00000000\_000000Salamanca

000004\_00000000\_000000000Badger

000005\_00000000\_00M Ehrmantraut

000006\_00000000\_00Lydia Rodarte

111111\_00000000\_000Walter White

222222\_00000000\_00Jesse Pinkman

333333\_00000000\_000Saul Goodman

444444\_00000000\_000Skylar White

555555\_00000000\_00Hank Schrader

666666\_00000000\_00000Heisenberg

777777\_00000000\_0000000Walt Jr.

888888\_00000000\_000000Gus Fring

999999\_00000000\_0000Skinny Pete

**Master Accounts Day 2**

000001\_00010000\_000Steven Gomez

000002\_00010000\_00000000Tortuga

000003\_00010000\_000000Salamanca

000004\_00010000\_000000000Badger

000005\_00010000\_00M Ehrmantraut

000006\_00010000\_00Lydia Rodarte

111111\_00005000\_000Walter White

222222\_00015000\_00Jesse Pinkman

333333\_00005000\_000Saul Goodman

444444\_00015000\_000Skylar White

555555\_00005000\_00Hank Schrader

666666\_00015000\_00000Heisenberg

777777\_00009900\_0000000Walt Jr.

888888\_00009900\_000000Gus Fring

999999\_00009900\_0000Skinny Pete

**Master Accounts Day 3**

000001\_00040000\_000Steven Gomez

000002\_00040000\_00000000Tortuga

000003\_00028000\_000000Salamanca

000004\_00039000\_000000000Badger

111111\_00005000\_000Walter White

222222\_00015000\_00Jesse Pinkman

333333\_00005000\_000Saul Goodman

444444\_00015000\_000Skylar White

555555\_00005000\_00Hank Schrader

666666\_00015000\_00000Heisenberg

777777\_00009900\_0000000Walt Jr.

888888\_00009900\_000000Gus Fring

999999\_00009900\_0000Skinny Pete

**Master Accounts Day 4**

000001\_00040000\_000Steven Gomez

000002\_00040000\_00000000Tortuga

000003\_00028000\_000000Salamanca

000004\_00039000\_000000000Badger

111111\_00005500\_000Walter White

222222\_00013950\_00Jesse Pinkman

333333\_00004000\_000Saul Goodman

444444\_00016000\_000Skylar White

555555\_00012500\_00Hank Schrader

666666\_00016500\_00000Heisenberg

777777\_00009900\_0000000Walt Jr.

888888\_00009900\_000000Gus Fring

999999\_00009900\_0000Skinny Pete

**Master Accounts Day 5**

000001\_00040100\_000Steven Gomez

000002\_00035000\_00000000Tortuga

000003\_00033000\_000000Salamanca

000004\_00040000\_000000000Badger

111111\_00005300\_000Walter White

222222\_00013950\_00Jesse Pinkman

333333\_00005000\_000Saul Goodman

444444\_00017000\_000Skylar White

555555\_00012400\_00Hank Schrader

666666\_00016400\_00000Heisenberg

777777\_00009900\_0000000Walt Jr.

888888\_00009900\_000000Gus Fring

999999\_00009900\_0000Skinny Pete

**Integration Report**

|  |  |
| --- | --- |
| Error | How It was fixed |
| Daily script did not merge transaction summary files | Changed daily script, cat command was misused, overcomplicated |
| Time stamped transaction summary files, being over written, by following session, due to them running so fast after each other, stamps as same time | Merged transaction summary files earlier to prevent this error. |
| Incorrect input for sessions | Go through session txt files and fix them |
| If master accounts is empty we get error in back end on Day 1 | Fixed by adding 00 to file at start, so that it can read in a value, and gets overwritten later |
| Session1-3.txt gets EOF FILE Error, didn’t stop at read | re-wrote txt file |
| Syntax error in back-end line 61 | accct[0] 🡪 acct[0] |
| Syntax error in back-end line 63, tries to compare in if statement with a list | range(list) 🡪 len(list) |
| Syntax error line 67-69 accNum not defined | accNum 🡪 acctNum |
| Syntax error line 109, writes new masteraccts but empty | Error found in backend and fixed |
| When creating accounts, account was added to master accounts multiple times | Rewrote create section |
| Index out of range line 64 | After multiple errors when creating accounts, that portion was rewritten |
| Valid accounts did not write correct output | Rewrote function |

**Front End**

import datetime

import time

import os.path

############################################ RETAIL #################################################

class retail(object):

def \_\_init\_\_(self, type,dailylimit):

self.type = type

self.dailylimit = dailylimit

def withdraw(self):

accNumInput = True

while (accNumInput):

accNum = int(raw\_input('Account Number: '))

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Withdrawal Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 100000):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

elif (self.dailylimit + amount > 100000):

print "This amount exceeds your daily limit."

else:

self.dailylimit += amount

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '02\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('02', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def deposit(self):

accNumInput = True

while (accNumInput):

accNum = raw\_input('Account Number: ')

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Deposit Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 100000):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

else:

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '01\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('01', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def transfer(self):

accNumInput = True

accNumInput2 = True

while(accNumInput):

accNumTo = raw\_input('To Account Number: ')

print str(accNumTo) + "\n"

#CHECK to SEE IF FIRST ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumTo)):

while (accNumInput2):

accNumFrom = raw\_input('From Account Number: ')

print str(accNumFrom) + "\n"

#CHECK TO SEE IF SECOND ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumFrom)):

accNumInput = False

accNumInput2 = False

amt = True

while (amt):

amount = int(input('Transfer Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 100000):

print "Please enter a valid transfer amount."

elif (amount < 0):

print "Please enter a valid transfer amount."

else:

amt = False

#create string for write file

accNumTo = str(accNumTo)

accNumFrom = str(accNumFrom)

amount = str(amount)

#transactionInfo = '03\_' + accNumTo + '\_' + accNumFrom + '\_' + amount

transactionInfo = formatFileLine('01', accNumTo, accNumFrom, amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

return transactionInfo

#METHOD WHICH RUNS ANY TRANSACTIONS FOR A RETAIL DAY

#WILL WRITE ANY TRANSACTIONS TO FILE

#LOGOUT IS ACCEPTED AT THIS STAGE

def runRetailDay(self):

running = True

#CREATES TRANSACTION SUMMARY FILE

ts = time.time()

st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

save\_path = '../SummaryFiles/'

file1 = 'Trans\_Summary\_File\_\_' + st + '.txt'

filename = file1.replace(":", "\_")

completeName = os.path.join(save\_path, filename)

f = open(completeName,'w')

#file1 = 'tempsummfile.txt'

#f = open(file1,'w')

while (running):

#STARTS ACCEPTING RETAIL TRANSACTIONS

transaction = raw\_input('Perform a transaction: ')

transaction.lower()

print str(transaction) + "\n"

#TESTS INPUT FOR WHICH TRANSACTION TYPE TO PERFORM

if (transaction == "withdraw"):

newTrans = self.withdraw()

f.write(newTrans + '\n')

elif (transaction == "deposit"):

newTrans = self.deposit()

f.write(newTrans + '\n')

elif (transaction == "transfer"):

newTrans = self.transfer()

f.write(newTrans + '\n')

elif (transaction == "logout"):

f.close()

running = False

else:

print "Please enter a valid transaction type."

return False

###########################################################################################################

############################################ AGENT #################################################

class agent(object):

def \_\_init\_\_(self, type):

self.type = type

def withdraw(self):

accNumInput = True

while (accNumInput):

accNum = raw\_input('Account Number: ')

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Withdrawal Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 999999):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

else:

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '02\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('02', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def deposit(self):

accNumInput = True

while (accNumInput):

accNum = raw\_input('Account Number: ')

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Deposit Amount (Cents): '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 999999):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

else:

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '01\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('01', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def transfer(self):

accNumInput = True

accNumInput2 = True

while(accNumInput):

accNumTo = raw\_input('To Account Number: ')

print str(accNumTo) + "\n"

#CHECK to SEE IF FIRST ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumTo)):

while (accNumInput2):

accNumFrom = raw\_input('From Account Number: ')

print str(accNumFrom) + "\n"

#CHECK TO SEE IF SECOND ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumFrom)):

accNumInput = False

accNumInput2 = False

amt = True

while (amt):

amount = int(raw\_input('Transfer Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 999999):

print "Please enter a valid transfer amount."

elif (amount < 0):

print "Please enter a valid transfer amount."

else:

amt = False

#create string for write file

accNumTo = str(accNumTo)

accNumFrom = str(accNumFrom)

amount = str(amount)

#transactionInfo = '03\_' + accNumTo + '\_' + accNumFrom + '\_' + amount

transactionInfo = formatFileLine('03', accNumTo, accNumFrom, amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

return transactionInfo

def create(self):

accNumInput = True

accNameInput = True

while (accNumInput):

accNum = int(input('Enter your desired account number: '))

print str(accNum) + "\n"

if ((len(str(accNum))) <= 6):

if (accNum <= 999999):

if (not acctNumExist(accNum)):

accNumInput = False

while (accNameInput):

accName = raw\_input('Enter your desired account name: ')

print str(accName) + "\n"

if (len(accName) > 15):

print "Please enter a valid account name."

elif (len(accName) == 0):

print "Please enter a valid account name."

else:

#create account number here

accNameInput = False

#create string for write file

accNum = str(accNum)

accName = str(accName)

#transactionInfo = '04\_' + accNum + "\_" + accName #proper formatting on end of string is needed

transactionInfo = formatFileLine('04', accNum, 'BBBBBB', 'MMMMMMMM', accName)

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

return transactionInfo

def delete(self):

accNumInput = True

accNameInput = True

while (accNumInput):

accNum = int(input('Enter the account number: '))

print str(accNum) + "\n"

#CHECK TO SEE IF INPUT ACCOUNT NUMBER EXISTS

if (acctNumExist(accNum)):

accNumInput = False

while (accNameInput):

accName = raw\_input('Enter the account name: ')

print str(accName) + "\n"

#CHECK TO SEE IF INPUT ACCOUNT NAME MATCHES ACCOUNT NUMBER

if (1 == 0): #backend thing

print "Please enter the proper account name for this account."

else:

#delete account now

accNameInput = False

#create string for write file

accNum = str(accNum)

accName = str(accName)

#transactionInfo = '05\_' + accNum + '\_' + accName #proper formatting on end of string is needed

transactionInfo = formatFileLine('05', accNum, 'BBBBBB', 'MMMMMMMM', accName)

else:

print "Please enter a valid account number."

return transactionInfo

#METHOD WHICH RUNS ANY TRANSACTIONS FOR A RETAIL DAY

#WILL WRITE ANY TRANSACTIONS TO FILE

#LOGOUT IS ACCEPTED AT THIS STAGE

def runAgentDay(self):

running = True

#CREATES TRANSACTION SUMMARY FILE

ts = time.time()

st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

save\_path = '../SummaryFiles/'

file1 = 'Trans\_Summary\_File\_\_' + st + '.txt'

filename = file1.replace(":", "\_")

completeName = os.path.join(save\_path, filename)

f = open(completeName,'w')

#file1 = 'tempsummfile.txt'

#f = open(file1,'w')

while (running):

#STARTS ACCEPTING RETAIL TRANSACTIONS

transaction = raw\_input('Perform a transaction: ')

transaction.lower()

print str(transaction) + "\n"

#TESTS INPUT FOR WHICH TRANSACTION TYPE TO PERFORM

if (transaction == "withdraw"):

newTrans = self.withdraw()

f.write(newTrans + '\n')

elif (transaction == "deposit"):

newTrans = self.deposit()

f.write(newTrans + '\n')

elif (transaction == "transfer"):

newTrans = self.transfer()

f.write(newTrans + '\n')

elif (transaction == "create"):

newTrans = self.create()

f.write(newTrans + '\n')

elif (transaction == "delete"):

newTrans = self.delete()

f.write(newTrans + '\n')

elif (transaction == "logout"):

f.write('00\n')

f.close()

running = False

else:

print "Please enter a valid transaction type."

return False

###########################################################################################################

def formatFileLine(transCode, firstAcctNum, secondAcctNum, acctAmt, acctName):

transCode = str(transCode)

firstAcctNum = str(firstAcctNum)

secondAcctNum = str(secondAcctNum)

acctAmt = str(acctAmt)

acctName = str(acctName)

#transaction code

if (len(transCode) == 2):

fileLine = transCode + "\_" #line: CC\_

#first account number

if (len(firstAcctNum) == 6):

firstAcctNum += "\_"

fileLine += firstAcctNum #line: CC\_AAAAAA\_

elif (len(firstAcctNum) < 6 and len(firstAcctNum) > 0): #pads 0 to beginning of account numbers

acctLength = len(firstAcctNum)

diff = 6 - acctLength

for i in range(diff):

firstAcctNum = "0" + firstAcctNum

firstAcctNum += "\_"

fileLine += firstAcctNum

#second account number

if (len(secondAcctNum) == 6):

secondAcctNum += "\_"

fileLine += secondAcctNum #line: CC\_AAAAAA\_BBBBBB\_

elif (len(secondAcctNum) < 6 and len(secondAcctNum) > 0):

acctLength = len(secondAcctNum)

diff = 6 - acctLength

for i in range(diff):

secondAcctNum = "0" + secondAcctNum

secondAcctNum += "\_"

fileLine += secondAcctNum

#transaction amount

if (len(acctAmt) == 8):

acctAmt += "\_"

fileLine += acctAmt #line: CC\_AAAAAA\_BBBBBB\_MMMMMMMM\_

elif (len(acctAmt) < 8 and len(acctAmt) > 0):

amtLength = len(acctAmt)

diff = 8 - amtLength

for i in range(diff):

acctAmt = "0" + acctAmt

acctAmt += "\_"

fileLine += acctAmt

#account name

if (len(acctName) == 15):

fileLine += acctName #line: CC\_AAAAAA\_BBBBBB\_MMMMMMMM\_NNNNNNNNNNNNNNN

elif (len(acctName) > 15):

newAcctName = ""

while (len(newAcctName) < 15):

for char in acctName:

newAcctName += char

fileLine += newAcctName

else:

nameLength = len(acctName)

diff = 15 - nameLength

for i in range(diff):

acctName = "0" + acctName

fileLine += acctName

return fileLine

def readAcctFile():

list = []

f = open('../validaccounts.txt')

list = f.readlines()

for x in range(len(list)):

list[x] = list[x].strip()

list[x] = int(list[x])

f.close()

return list

def acctNumExist(num):

num = int(num)

for x in accounts:

if (x == num):

return True

return False

def openBankingSystem():

loggedIn = True

while (loggedIn):

#GETS LOGIN TO START, STAGE 0

firstInput = raw\_input('Type "login" to login: ')

firstInput.lower()

firstInput = str(firstInput)

print firstInput + "\n"

if (firstInput == "login"):

pickDay = True

while (pickDay):

#ACCEPTS INPUT FOR AGENT OR RETAIL, STAGE 1

dayType = raw\_input('agent or retail: ')

dayType.lower()

dayType = str(dayType)

print dayType + "\n"

if (dayType == "retail"):

pickDay = False

retailDay = retail(dayType,0)

loggedIn = retailDay.runRetailDay()

elif (dayType == "agent"):

pickDay = False

agentDay = agent(dayType)

loggedIn = agentDay.runAgentDay()

else:

print "Please enter a valid input.\n"

elif (firstInput == "stop"):

#Entering stop will kill the program

return 0

else:

print "Please enter a valid input.\n"

#STARTS OVER AGAIN AFTER LOGOUT AT STAGE 0

return openBankingSystem()

###### MAIN PROGRAM ######

accounts = readAcctFile()

openBankingSystem()

**Back End**

import sys

def transaction(masterAccts,trans):

#take trans, split by \_ into list

transCopy = trans.split('\_') #[CC, AAAAAA, BBBBBB, MMMMMMMM, NNNNNNNNNNNNNNN]

master = []

for i in range(len(masterAccts)):

master.append(masterAccts[i])

for i in range(len(master)):

master[i] = master[i].split('\_')

if (transCopy[0] == '01'): #deposit

for acct in range(len(masterAccts)):

if (master[acct][0] == transCopy[1]):

acctBalance = int(master[acct][1])

depAmount = int(transCopy[3])

acctBalance += depAmount

master[acct][1] = str(master[acct][1])

master[acct][1] = str(acctBalance)

newStr = format(master[acct][0], master[acct][1], master[acct][2])

masterAccts[acct] = newStr

return masterAccts

#withdraw

elif (transCopy[0] == '02'):

for acct in range(len(masterAccts)):

if (master[acct][0] == transCopy[1]):

acctBalance = int(master[acct][1])

depAmount = int(transCopy[3])

acctBalance -= depAmount

master[acct][1] = str(master[acct][1])

master[acct][1] = str(acctBalance)

newStr = format(master[acct][0], master[acct][1], master[acct][2])

masterAccts[acct] = newStr

return masterAccts

elif (transCopy[0] == '03'): #transfer

for acct in range(len(masterAccts)):

if (master[acct][0] == transCopy[1]):

for anotherAcct in range(len(masterAccts)):

if (master[anotherAcct][0] == transCopy[2]):

recAcctBalance = int(master[acct][1])

transAcctBalance = int(master[anotherAcct][1])

transAmt = int(transCopy[3])

recAcctBalance += transAmt

transAcctBalance -= transAmt

master[acct][1] = str(master[acct][1])

master[anotherAcct][1] = str(master[acct][1])

newStrFirstAcct = format(master[acct][0], master[acct][1], master[acct][2])

masterAccts[acct] = newStr

newStr = format(master[anotherAcct][0], master[anotherAcct][1], master[anotherAcct][2])

masterAccts[anotherAcct] = newStr

return masterAccts

elif (transCopy[0] == '04'):

acctNum = int(transCopy[1])

newStr = format(transCopy[1], "00000000", transCopy[4])

for acct in range(len(master)):

if (master[acct][0] == acctNum):

throwError()

else:

if (master[acct][0] == ''):

skip = 0

else:

current = int(master[acct][0])

if (acctNum < current):

masterAccts.insert(acct, newStr)

return masterAccts

masterAccts.append(newStr)

return masterAccts

elif (transCopy[0] == '05'): #delete \_ do decision testing, need a test case it evaluate every if both ways

acctNum = str(transCopy[1])

transAcctName = str(transCopy[4])

for acct in range(len(master)):

if (acctNum == master[acct][0]):

acctBalance = master[acct][1]

if (acctBalance == '00000000'):

acctName = str(master[acct][2])

if (transAcctName == acctName):

masterAccts = masterAccts.remove(masterAccts[acct])

else:

throwError()

else:

throwError()

return masterAccts

elif (transCopy[0] == '00'):

return masterAccts

def format(num, balance, name):

string = str(num) + "\_" + str(balance) + "\_" + str(name)

return string

def writeNewMasterAccounts(list):

f = open('./masteraccounts.txt','w')

for i in list:

f.write(i + "\n")

f.close()

return 0

def writeNewValidAccounts(list):

f = open('./validaccounts.txt','w')

valid = []

for i in range(len(list)):

valid.append(list[i])

for i in range(len(valid)):

valid[i] = valid[i].split('\_')

for i in range(len(list)):

f.write(valid[i][0] + "\n")

f.write("000000")

f.close()

return 0

def throwError():

sys.exit('Fatal Error')

def main\_program():

#open master accounts

masteraccts = []

f = open('./masteraccounts.txt')

masteraccts = f.readlines()

for x in range(len(masteraccts)):

masteraccts[x] = masteraccts[x].strip()

f.close()

#open merged transaction file

mergedtrans= []

f = open('./mergedtransactions.txt', 'r')

mergedtrans = f.readlines()

for x in range(len(mergedtrans)):

mergedtrans[x] = mergedtrans[x].strip()

f.close()

#for all transactions update the master accounts file

for i in mergedtrans:

masteraccts = transaction(masteraccts,i)

writeNewValidAccounts(masteraccts)

writeNewMasterAccounts(masteraccts)

return 0

main\_program()