Done by Chinese-broccoli (Mason Chang, Yueyang Liu & Ruikang Luo)

Daily.py

```
import frontEnd as fn
import backEnd as bn
import sys
import os

class daily:
    def __init__(self, day):
        self.day = "day" + str(day)                                             # processing day correspon
ding to a directory
        self.rootDir = os.path.join(os.path.dirname(__file__), self.day)        # all files read and writt
en within the directory
        self.merge_file_from = os.path.join(self.rootDir, "frontend_out")       # merge files from fronten
d output
        self.merge_file_to = os.path.join(self.rootDir, "backend_in")           # merged file as backend i
nput
        name = "mergedSummaryFile.txt"                                          # the merged file name
        self.open_frontend()
        self.write_file(name)
        self.pass_backend(name)

    # run the fronted end
    def open_frontend(self):
        write_to = os.path.join(self.merge_file_from, "transaction.txt")        # frontend ouput file writ
e to
        trans = os.path.join(self.rootDir, "frontend_in/trans.txt")             # user input come from the
 file
        account_list = os.path.join(self.rootDir, "frontend_in/account.txt")    # same day, the same accou
nt list
        for i in range(1,4):                                                    # 1, 2, 3 sessions
            write_to_temp = os.path.splitext(write_to)[0] + str(i) + ".txt"
            trans_temp = os.path.splitext(trans)[0] + str(i) + ".txt"
            sys.argv = ["frontEnd.py", account_list, write_to_temp]
            f= open(trans_temp, "r")
            sys.stdin = f
            fn.main()
            f.close()

    # pass backend
    def pass_backend(self, name):
        backend_inputDir = self.merge_file_to
        masterAcc = os.path.join(backend_inputDir, "masterAccount.txt")
        mergeTrans = os.path.join(backend_inputDir, name)
```

```python
        sys.argv = ['backEnd.py', mergeTrans, masterAcc]
        bn.main()

    # write the merged file
    def write_file(self, name):
        mergered_file = self.read_merge_file(self.merge_file_from)
        targetFile = os.path.join(self.merge_file_to, name)
        if not os.path.exists(self.merge_file_to):
            os.makedirs(self.merge_file_to)
        with open(targetFile, "w") as f:
            f.writelines(mergered_file)

    # Read files and merge them
    def read_merge_file(self, targetDir):
        flist = []
        for filename in os.listdir(targetDir):
            if filename.endswith('.txt'):
                with open(os.path.join(targetDir,filename)) as f:
                    flist += f.readlines()[:-1]
        flist.append("EOS")
        return flist

def main():
    daily(1)

if __name__ == "__main__":
    main()
```

Weekly.py

```python
import Daily as dy
import os


class Weekly:
    def __init__(self):
        for i in range(1,6):
            print('Welcome to DAY ' + str(i))
            dy.daily(i)
            if i != 5:
                self.updateFile(i)
                print('\n' * 5)

    # prepare next day's account list and master account
    def updateFile(self, d):
        f1 = open("day"+ str(d) +"/backend_out/masterAccount.txt", 'r')     # today's back end output is n
ext
        f2 = open("day"+ str(d+1) + "/backend_in/masterAccount.txt", 'w')   # day back end input
        m = f1.readlines()
        for item in m:
            f2.write(item)
        f1.close()
        f2.close()
        f1 = open("day"+ str(d) +"/backend_out/accountList.txt", 'r')
        f2 = open("day"+ str(d+1) + "/frontend_in/account.txt", 'w')
        m = f1.readlines()
        for item in m:
            f2.write(item)
        f1.close()
        f2.close()

def main():
    Weekly()

if __name__ == "__main__":
    main()
```

printout of each front end:

	Transaction input	Transaction output
Front end #1	login agent createacct 1000327 newAcc01 createacct 1000330 newAcc04 logout	NEW 1000327 000 0000000 newAcc01 NEW 1000330 000 0000000 newAcc04 EOS
Front end #2	login agent createacct 1000328 newAcc02 createacct 1000332 newAcc06 logout	NEW 1000328 000 0000000 newAcc02 NEW 1000332 000 0000000 newAcc06 EOS
Front end #3	login agent	NEW 1000329 000 0000000 newAcc03 NEW 1000331 000 0000000 newAcc05

	createacct 1000329 newAcc03 createacct 1000331 newAcc05 logout	EOS

Merged transaction:

NEW 1000327 000 0000000 newAcc01

NEW 1000330 000 0000000 newAcc04

NEW 1000328 000 0000000 newAcc02

NEW 1000332 000 0000000 newAcc06

NEW 1000329 000 0000000 newAcc03

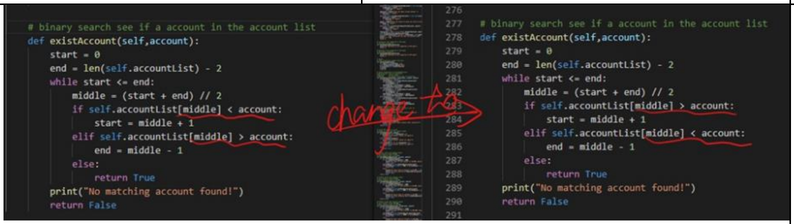
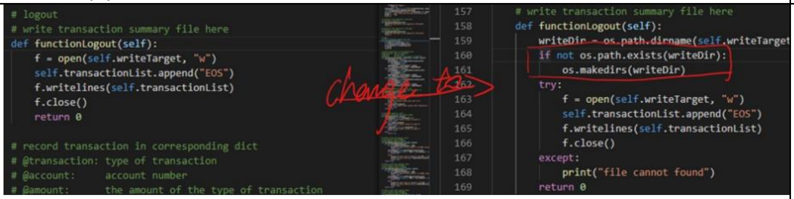
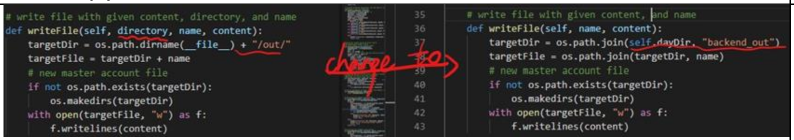
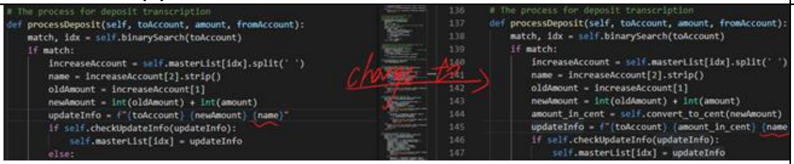
NEW 1000331 000 0000000 newAcc05

EOS

Printout of master account:

DAY	in	out
Day1	Empty	1000332 000 newAcc06 1000331 000 newAcc05 1000330 000 newAcc04 1000329 000 newAcc03 1000328 000 newAcc02 1000327 000 newAcc01
Day2	1000332 000 newAcc06 1000331 000 newAcc05 1000330 000 newAcc04 1000329 000 newAcc03 1000328 000 newAcc02 1000327 000 newAcc01	1000332 000 newAcc06 1000331 50000 newAcc05 1000330 50000 newAcc04 1000329 50000 newAcc03 1000328 50000 newAcc02 1000327 50000 newAcc01
Day3	1000332 000 newAcc06 1000331 50000 newAcc05 1000330 50000 newAcc04 1000329 50000 newAcc03 1000328 50000 newAcc02 1000327 50000 newAcc01	1000331 35000 newAcc05 1000330 650000 newAcc04 1000329 50000 newAcc03 1000328 75000 newAcc02 1000327 25000 newAcc01 1000326 000 newAcc07
Day4	1000331 35000 newAcc05 1000330 650000 newAcc04 1000329 50000 newAcc03	1000338 000 newAcc038 1000331 585000 newAcc05 1000330 100000 newAcc04

	1000328 75000 newAcc02 1000327 25000 newAcc01 1000326 000 newAcc07	1000329 50000 newAcc03 1000328 65000 newAcc02 1000327 5025000 newAcc01 1000326 94999999 newAcc07
Day5	1000338 000 newAcc038 1000331 585000 newAcc05 1000330 100000 newAcc04 1000329 50000 newAcc03 1000328 65000 newAcc02 1000327 5025000 newAcc01 1000326 94999999 newAcc07	1000338 4999999 newAcc038 1000331 775000 newAcc05 1000329 150000 newAcc03 1000328 65000 newAcc02 1000327 5000000 newAcc01 1000326 90000000 newAcc07

Defect	Cause	Fixed	Comment
Can not find exist account in account list	 In atm.py		The order of account list was not matching. Now change it to descending order
Write file failed when log out	 In atm.py		Check the writing directory if exist. If not create the directory first
Writing directory was not expected	 In backend.py		Change the backend output to "backend_out". More logically make senses
Output masterlist had wrong format	 In backend.py		Add a new line character to the update info
Program would output "2" when int("001")+int("001")	See code below		Add "0" to the result if it necessary.

Row 1, Cause:

```python
# binary search see if a account in the account list
def existAccount(self,account):
    start = 0
    end = len(self.accountList) - 2
    while start <= end:
        middle = (start + end) // 2
        if self.accountList[middle] < account:
            start = middle + 1
        elif self.accountList[middle] > account:
            end = middle - 1
        else:
            return True
    print("No matching account found!")
    return False
```

Row 1, Fixed:

```python
# binary search see if a account in the account list
def existAccount(self,account):
    start = 0
    end = len(self.accountList) - 2
    while start <= end:
        middle = (start + end) // 2
        if self.accountList[middle] > account:
            start = middle + 1
        elif self.accountList[middle] < account:
            end = middle - 1
        else:
            return True
    print("No matching account found!")
    return False
```

Row 2, Cause:

```python
# logout
# write transaction summary file here
def functionLogout(self):
    f = open(self.writeTarget, "w")
    self.transactionList.append("EOS")
    f.writelines(self.transactionList)
    f.close()
    return 0

# record transaction in corresponding dict
# @transaction: type of transaction
# @account:     account number
# @amount:      the amount of the type of transaction
```

Row 2, Fixed:

```python
    # write transaction summary file here
    def functionLogout(self):
        writeDir = os.path.dirname(self.writeTarget
        if not os.path.exists(writeDir):
            os.makedirs(writeDir)
        try:
            f = open(self.writeTarget, "w")
            self.transactionList.append("EOS")
            f.writelines(self.transactionList)
            f.close()
        except:
            print("file cannot found")
        return 0
```

Row 3, Cause:

```python
# write file with given content, directory, and name
def writeFile(self, directory, name, content):
    targetDir = os.path.dirname(__file__) + "/out/"
    targetFile = targetDir + name
    # new master account file
    if not os.path.exists(targetDir):
        os.makedirs(targetDir)
    with open(targetFile, "w") as f:
        f.writelines(content)
```

Row 3, Fixed:

```python
# write file with given content, and name
def writeFile(self, name, content):
    targetDir = os.path.join(self.dayDir, "backend_out")
    targetFile = os.path.join(targetDir, name)
    # new master account file
    if not os.path.exists(targetDir):
        os.makedirs(targetDir)
    with open(targetFile, "w") as f:
        f.writelines(content)
```

Row 4, Cause:

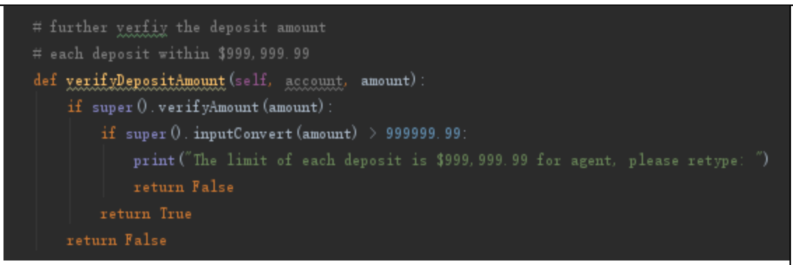
```python
# The process for deposit transcription
def processDeposit(self, toAccount, amount, fromAccount):
    match, idx = self.binarySearch(toAccount)
    if match:
        increaseAccount = self.masterList[idx].split(' ')
        name = increaseAccount[2].strip()
        oldAmount = increaseAccount[1]
        newAmount = int(oldAmount) + int(amount)
        updateInfo = f"{toAccount} {newAmount} {name}"
        if self.checkUpdateInfo(updateInfo):
            self.masterList[idx] = updateInfo
        else:
```

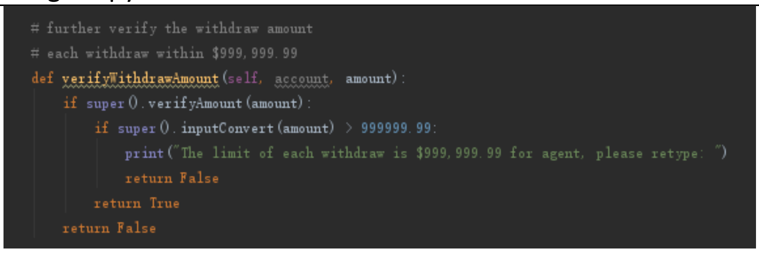
Row 4, Fixed:

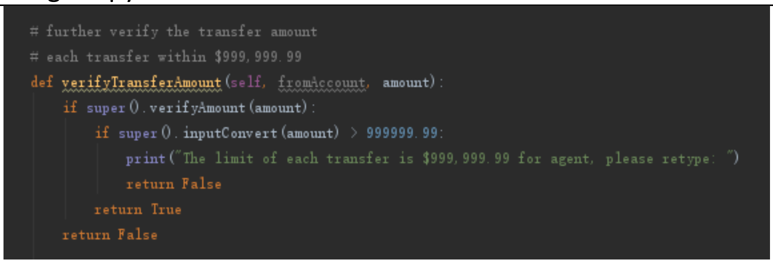
```python
# The process for deposit transcription
def processDeposit(self, toAccount, amount, fromAccount):
    match, idx = self.binarySearch(toAccount)
    if match:
        increaseAccount = self.masterList[idx].split(' ')
        name = increaseAccount[2].strip()
        oldAmount = increaseAccount[1]
        newAmount = int(oldAmount) + int(amount)
        amount_in_cent = self.convert_to_cent(newAmount)
        updateInfo = f"{toAccount} {amount_in_cent} {name
        if self.checkUpdateInfo(updateInfo):
            self.masterList[idx] = updateInfo
```

Row 5, Cause:

```python
def convert_to_cent(self, num):
    amount_in_cent = str(num)
    if (num < 10):
        amount_in_cent = "00" + amount_in_cent
    elif (num < 100):
        amount_in_cent = "0" + amount_in_cent
    return amount_in_cent
```

Failed to deposit more than $2,000 in agent mode	
	`# further verfiy the deposit amount`
	`# each deposit within $999,999.99`
	`def verifyDepositAmount(self, account, amount):`
	`    if super().verifyAmount(amount):`
	`        if super().inputConvert(amount) > 999999.99:`
	`            print("The limit of each deposit is $999,999.99 for agent, please retype: ")`
	`            return False`
	`        return True`
	`    return False`
	In agent.py
		Overwrite the verifyDepositAmount function (Omission when design test case)

Failed to withdraw more than $1,000 in agent mode	
	`# further verify the withdraw amount`
	`# each withdraw within $999,999.99`
	`def verifyWithdrawAmount(self, account, amount):`
	`    if super().verifyAmount(amount):`
	`        if super().inputConvert(amount) > 999999.99:`
	`            print("The limit of each withdraw is $999,999.99 for agent, please retype: ")`
	`            return False`
	`        return True`
	`    return False`
	In agent.py
		Overwrite the verifyWithdrawAmount function (Omission when design test case)

Failed to transfer more than $10,000 in agent mode	
	`# further verify the transfer amount`
	`# each transfer within $999,999.99`
	`def verifyTransferAmount(self, fromAccount, amount):`
	`    if super().verifyAmount(amount):`
	`        if super().inputConvert(amount) > 999999.99:`
	`            print("The limit of each transfer is $999,999.99 for agent, please retype: ")`
	`            return False`
	`        return True`
	`    return False`
	In agent.py
		Overwrite the verifyTransferAmount function in (Omission when design test case)