Lab 2:
You are to write a program that reads in lines of input using scanf or cin, and file redirection, process the data, and then out the results using printf or cout. This means you can use C, C++, Java, Php, Python or any other style I/O as is your personal preference.

- Each line of the data file begins with an 'e', 'c', or 't'.
- Lines beginning with 'e' will have an integer employee ID and an employee name.
- Lines beginning with 'c' will have an integer customer ID, customer name, and a floating-point account balance.
- Lines beginning with a 't' will have a customer ID, employee ID, a 'w' or 'd' representing withdrawal or deposit, and a floating-point transaction amount.
- Lines beginning with an 'e' and 'c' may be intermingled, but all 'e' and 'c' lines will come before lines beginning with 't'.
- There will be at most 50 employees and 50 customers, and an unknown number of transactions. Names will be at most 15 characters long.

Plan.txt

My Program will have:
Dictionary for:
Customer
Employee
Transactions
Log

Methods like:
Read File method:
read the content of the data file.
Create Customer Dictionary method:
Customer dictionary creation with the data read from the file.
Create Employee Dictionary method:
Employee dictionary creation with the data read from the file.
Create Transaction Dictionary method:
Transaction dictionary creation with the data read from the file.
Deposit Function:
Read the details of customer and Employee
Add the deposited balance to the previous balance
log the changes
Withdraw Function
Read the details of customer and Employee
Deduct the withdrawn balance to the previous balance
log the changes

Transaction function
Process deposit and withdraw
Create Log File
Transaction Logs.

Task, Time, Line of Codes
Define dictionary, 5, 3
Read File method,3,3
Create Employee Dictionary method, 15, 20
Create Customer Dictionary method, 15, 20
Create Transaction Dictionary method, 15, 20
Transaction function, 10, 8
Deposit function, 15, 20
Withdraw function, 15, 20
Create log file, 10, 10

Task.log

Timestamp	**Task**	**Name**	**Duration**	**Lines of Code**
10/08/2020 1:13	Coding	MainMethod	4	11
10/08/2020 1:17	Coding	ReadFile	3	2
10/08/2020 1:19	Coding	Employee_Dictionary	10	5
10/08/2020 1:29	Coding	Customer_Dictionary	7	7
10/08/2020 1:36	Coding	Transaction_Dictionary	6	9
10/08/2020 1:40	Coding	Deposit_Function	15	4
10/08/2020 1:55	Coding	Withdraw_Function	5	4
10/08/2020 2:00	Coding	Transaction_Function	2	6
10/08/2020 2:02	Coding	Transaction_log	5	3
10/08/2020 2:07	Coding	Error Fixing	25	

Error.log

Timestamp	Error
10/08/2020 2:10	Undefined variable 'filename'
10/08/2020 2:12	Undefined variable 'line'
10/08/2020 2:12	Undefined variable 'info_transaction'
10/08/2020 2:17	ValueError: Invalid format specifier

Data File

e 5 Elden
c 3 Felipe 55342.51415
e 3 Leonardo
e 1 Yong
c 9 Alessandra 8114.541862
c 6 Marnie 15287.78233
e 8 Kourtney
c 2 Lou 95053.44742
c 5 Numbers 51245.66138
e 4 Jarvis
e 9 Marlen
e 10 Florance
c 1 Devon 56442.27875
e 2 Elliott
c 8 Justina 73723.84849
c 10 Reyna 82946.53205
e 6 Antonetta
e 7 Florene
c 4 Merrill 98281.82784
c 7 Marlana 33252.21805
t 1 8 w 4924.86
t 9 6 d 3220.42
t 6 1 w 127.62
t 9 8 w 5566.7
t 9 8 d 5414.55
t 5 7 w 9422.35

t 9 10 d 1382.07
t 4 7 d 6131.07
t 8 2 w 2362.22
t 8 10 d 5834.48
t 5 4 w 5150.73
t 6 2 d 3795.96
t 1 9 w 3919.45
t 5 2 w 5037.31
t 3 9 w 8129.21
t 8 6 d 1235.67
t 2 4 d 6901.28
t 8 10 d 5599.44
t 6 9 d 1936.16
t 7 9 d 7363.98

Correct Output(For testing)

12345678901234567890123456789012345678901234567890123456789012345678901234567890
Devon Kourtney -$4924.86 $ 51517.42
Alessandra Antonetta +$3220.42 $ 11334.96
Marnie Yong -$127.62 $ 15160.16
Alessandra Kourtney -$5566.70 $ 5768.26
Alessandra Kourtney +$5414.55 $ 11182.81
Numbers Florene -$9422.35 $ 41823.31
Alessandra Florance +$1382.07 $ 12564.88
Merrill Florene +$6131.07 $ 104412.90
Justina Elliott -$2362.22 $ 71361.63
Justina Florance +$5834.48 $ 77196.11
Numbers Jarvis -$5150.73 $ 36672.58
Marnie Elliott +$3795.96 $ 18956.12
Devon Marlen -$3919.45 $ 47597.97
Numbers Elliott -$5037.31 $ 31635.27
Felipe Marlen -$8129.21 $ 47213.30
Justina Antonetta +$1235.67 $ 78431.78
Lou Jarvis +$6901.28 $ 101954.73
Justina Florance +$5599.44 $ 84031.22
Marnie Marlen +$1936.16 $ 20892.28
Marlana Marlen +$7363.98 $ 40616.20

Program:

```
info_employee = dict()
info_customer = dict()
detail_transaction = list()
log_transaction = list ()

def ReadFile(name: str):
    file= open(name, "r").readlines()
    return file

def Employee_Dictionary(filedata: list):
    for data in filedata:
        data= data.strip().split()
        if data[0] == "e":
            empid = data[1]
            info_employee[empid] = data[2]

def Customer_Dictionary(filedata: list):
    for data in filedata:
        data= data.strip().split()
        if data[0] == "c":
            cusid = data[1]
            info_customer[cusid] = dict()
            info_customer[cusid]["CusName"] = data[2]
            info_customer[cusid]["balance"] = float(data[3])

def Transaction_Dictionary(filedata: list):
    for data in filedata:
        data= data.strip().split()
        if data[0] == "t":
            data_transaction = dict()
            data_transaction["cusID"] = data[1]
            data_transaction["empID"] = data[2]
            data_transaction["transacType"] = data[3]
            data_transaction["amount"] = float(data[4])
            detail_transaction.append(data_transaction)

def Deposit(transaction: dict):
    customerName =
info_customer[transaction["cusID"]]["CusName"]
```

```
    employeeName = info_employee[transaction["empID"]]
    info_customer[transaction["cusID"]]["balance"] +=
transaction["amount"]
    log_transaction.append(f"{customerName} {employeeName}
+${transaction['amount']:.2f} $
{info_customer[transaction['cusID']]['balance']:.2f}\n")

def Withdraw(transaction: dict):
    customerName =
info_customer[transaction["cusID"]]["CusName"]
    employeeName = info_employee[transaction["empID"]]
    info_customer[transaction["cusID"]]["balance"] -=
transaction["amount"]
    log_transaction.append(f"{customerName} {employeeName}
-${transaction['amount']:.2f} $
{info_customer[transaction['cusID']]['balance']:.2f}\n")

def Transaction():

log_transaction.append("12345678901234567890123456789012345678901
12345678901234567890\n")
    for transaction in detail_transaction:
        if transaction["transacType"] == "d":
            Deposit(transaction)
        else:
            Withdraw(transaction)

def TransactionLog(filename: str):
    with open(filename, "w") as fhandler:
        fhandler.writelines(log_transaction)
        print("Transaction log successfully maintained !!!")

def TestResult(testData: list):
    for i in range(len(log_transaction)):
        if testData[i].strip() == log_transaction[i].strip():
            print(f"{i + 1}th data is verified.")
        else:
            print(f"{i + 1}th data is verified.")

filedata = ReadFile("file.txt")
```

```
Employee_Dictionary(filedata)
Customer_Dictionary(filedata)
Transaction_Dictionary(filedata)

Transaction()
testdata = ReadFile("CorrectOutput.txt")
TestResult(testdata)
TransactionLog("transactionLog.txt")
```

Output :

As generated on the transaction log file:

12345678901234567890123456789012345678901234567890123456789012345678901234567890
Devon Kourtney -$4924.86 $ 51517.42
Alessandra Antonetta +$3220.42 $ 11334.96
Marnie Yong -$127.62 $ 15160.16
Alessandra Kourtney -$5566.70 $ 5768.26
Alessandra Kourtney +$5414.55 $ 11182.81
Numbers Florene -$9422.35 $ 41823.31
Alessandra Florance +$1382.07 $ 12564.88
Merrill Florene +$6131.07 $ 104412.90
Justina Elliott -$2362.22 $ 71361.63
Justina Florance +$5834.48 $ 77196.11
Numbers Jarvis -$5150.73 $ 36672.58
Marnie Elliott +$3795.96 $ 18956.12
Devon Marlen -$3919.45 $ 47597.97
Numbers Elliott -$5037.31 $ 31635.27
Felipe Marlen -$8129.21 $ 47213.30
Justina Antonetta +$1235.67 $ 78431.78
Lou Jarvis +$6901.28 $ 101954.73
Justina Florance +$5599.44 $ 84031.22
Marnie Marlen +$1936.16 $ 20892.28
Marlana Marlen +$7363.98 $ 40616.20