Problem Statement

In this assignment, you will use the code you developed in Hack 14.0 to produce several additional reports on the financial transaction data used in that hack. Though you may have collaborated with others on the code in that hack, all code for this assignment must be your own as usual.

All of the reports below should be executed from the same main function as in the previous hack, so you will need to hand in all of your source files, `transaction.h`, `transaction.c`, and `transactionReport.c`. You should use/modify your test case from Hack 14.0 but you need not hand it in.

Validating the Data

A common fundamental operation of data processing is to validate data to ensure that it is consistent. For example, in our transaction data, the difference in the account balance before and after the transaction *should* be equal to the amount of the transaction.

Note that deposit transactions should *increase* the balance of the account while other transaction types *decrease* the balance of the account. Add to your program, code that produces a report of all invalid transactions. Your report should include a list of all invalid transactions as well as an indication of *why* they are invalid. Include a summary of the number of invalid transactions as well as a total of how much they are off by. An example is provided in Figure 1.

```
1  ==================================================
2  Balance Validation Report
3  ==================================================
4    Invalid Transaction, off by $0.01:
5          003C45B2-1356-4C7F-8D8A-EBB7BF026F16: C386278926  -> C1113941243
6          ( Deposit) $ 149504.20 ($56082.00 -> $205586.19)
7    Invalid Transaction, off by $103.00:
8          1DBB66AE-9222-4D3D-8651-BAB82CB48488: C1824038590 -> C621979846
9          ( Deposit) $ 270574.43 ($119.00 -> $270590.43)
10   Invalid Transaction, off by $-100.00:
11         1F5B3912-352E-412D-AEF7-4ED65BF38CE0: C327263195  -> C956086567
12         ( Deposit) $ 281292.06 ($17468.00 -> $298860.06)
13   Invalid Transaction, off by $0.01:
14         458BF06E-1E34-4E1B-9656-7D78032C70E0: C1325645367 -> M1735850501
15         ( Payment) $   4319.43 ($15463.00 -> $11143.58)
16 Number of invalid balances found: 4
```

Figure 1: Simple consistency report output example

Fraud Detection: Reporting Limits & Suspicious Limits

For tax purposes and as a tool to detect fraud and money laundering, federal law requires banks to report all transactions (especially deposits) of $10,000 or more to the IRS. Since this is widely known, individuals trying to hide activity from the IRS will typically make transactions that are just below this threshold.

Add code to your program to produce a report of *the number of* deposits in excess of $10,000 and, separately, the number of deposits whose amounts are between $9,900 and $9,999.99. An example report may look something like that in Figure 2

```
Deposit Limit Report
====================================
Deposits exceeding 10k:   482237
Deposits approaching 10k: 77
```

Figure 2: Simple limit report output example

Fraud Detection: Repeated Transactions

Another common red flag for money laundering activity is repeated transactions between the same accounts for the same amount. Add code to your program that identifies all transactions that have the same source/destination accounts in the same amount and print them all to the standard output. An example report may look something like that in Figure 3

```
================================================
Repeated Transaction Report
================================================
Repeated Transactions:
        4200730A-C0EE-4D87-8F89-B2DF063C3D7F: C1274943420 -> M697432348
        ( Payment) $  28277.99 ($39805.00 -> $11527.01)
        9200730A-C0EE-4D87-8F89-B2DF063C3D7F: C1274943420 -> M697432348
        ( Payment) $  28277.99 ($39805.00 -> $11527.01)
        0000730A-C0EE-4D87-8F89-B2DF063C3D7F: C1274943420 -> M697432348
        ( Payment) $  28277.99 ($39805.00 -> $11527.01)
Total repeated transactions flagged: 2
```

Figure 3: Simple repeated transaction report output example

Fraud Detection: Benford's Law

Another useful tool in financial forensics is *Benford's Law* which is an observation that naturally occurring figures follow a well-defined distribution. In particular, the most common first digit in a collection of numbers is typically 1, followed by 2 as the second most common and so on. In fact, for each digit n, $1 \leq n \leq 9$, the expectation that n is the first digit of a number is

$$\log_{10}\left(1 + \frac{1}{n}\right)$$

This gives us the expectations that can be found in Table 1.

Table 1: Frequency expectation of the first digit of numbers according to Benford's Law

Digit	Frequency
1	0.30103
2	0.17609
3	0.12494
4	0.09691
5	0.07918
6	0.06695
7	0.05799
8	0.05115
9	0.04576

One way of detecting fraud is to analyze a set of numbers to see if they follow the same expected distribution. If so, then they might be legitimate. If the first digit of the set of numbers does not follow Benford's law, then it may be an indication that they were generated by some artificial or random process.

Add code to your program to produce a report of the distribution of first digits in the amounts of all transactions. An example report may look something like that in Figure 4

Fraud Detection: Finding Data Anomalies

Databases are frequently backed up at regular intervals called "snapshots." Consider the following scenario: a financial auditor suspects that an employee has committed fraud by modifying transactions by modifying the amounts involved or the balances or the account numbers or even deleting transactions entirely. Legitimate transactions are usually "undone" by creating an additional transaction, not by removing them.

Help the auditor identify potential issues by comparing two database snapshots and finding any data anomalies between them. Given two database snapshots, you should

```
==================================================
Benford Analysis
==================================================
Number  Count   Frequency    Expected  Difference
1          33       0.330       0.301       0.029
2          18       0.180       0.176       0.004
3          12       0.120       0.125      -0.005
4          10       0.100       0.097       0.003
5           6       0.060       0.079      -0.019
6           5       0.050       0.067      -0.017
7           6       0.060       0.058       0.002
8           6       0.060       0.051       0.009
9           4       0.040       0.046      -0.006
```

Figure 4: Benford's Law report output example

find any transactions that disagree in any way (account numbers, amounts, balances, etc.) or any transactions that are in one snapshot but not the other.

Given two database dumps (one previous to the suspected fraud and one after) find any data anomolies: that is, missing transactions or "new" transactions and/or differences in amounts/balances that may have been modified

Add code to your program to produce a report that identifies all such data anomalies. For this report, you will need to accept *two* file names as command line arguments. An example report may look something like that in Figure 5

Note: the databases you will run reports on will be moderately large (millions of records). You will *not* be able to design a "brute-force" solution that tests every pair of transactions between the two files (which would be trillions of operations). Instead, you will need to be more clever in the solution you design.

```
Missing transaction:
  4836A6C0-FDFE-4107-8A7F-5BA8C941BB3E in database A but not in B
Inconsistent data:
        A: BC377639-37CC-4824-81AF-60177418B46D: C1906093041 -> M95867054
           ( Payment) $  14535.18 ($83310.00 -> $68774.82)
        B: BC377639-37CC-4824-81AF-60177418B46D: C1906093041 -> M95867054
           ( Payment) $  14530.18 ($83310.00 -> $68774.82)
Inconsistent data:
        B: BC377639-37CC-4824-81AF-60177418B46D: C1906093041 -> M95867054
           ( Payment) $  14530.18 ($83310.00 -> $68774.82)
        A: BC377639-37CC-4824-81AF-60177418B46D: C1906093041 -> M95867054
           ( Payment) $  14535.18 ($83310.00 -> $68774.82)
Missing transaction:
  C306CA7E-80A0-499E-8CDE-13DFB50C6753 in database B but not in A
Total missing records:      2
Total inconsistent records: 2
```

Figure 5: Report on data anomalies between two database snapshots.