Modified slightly from https://homework.adhoc.team/proto/

Parse a custom protocol format

A payment processing application you are working on must interface with an old-school mainframe format that we've named "MPS7". This means consuming a proprietary binary protocol format that no one on your team is familiar with yet.

Task

You must write a program that reads in a transaction log and parses it. The transaction log will be named `txnlog.dat` and be in the same directory as your program. A sample `txnlog.dat` file is provided for you. The log should be parsed according to the specification in the Notes section below. Your program must answer the following 5 questions:

- What is the total amount in dollars of credits?
- What is the total amount in dollars of debits?
- How many autopays were started?
- How many autopays were ended?
- What is balance of user ID 2456938384156277127?

Your program must output the answers in the format below. For example, if your program determined that the answer for each question was zero, your program would output:

```
total credit amount=0.00
total debit amount=0.00
autopays started=0
autopays ended=0
balance for user 2456938384156277127=0.00
```

You must supply your source code as part of your answer. Write your code in Java (preferred) or Python. We'll want to compile your code from source and run it from a Unix-like command line, so please include the complete instructions for doing so in a COMMENTS file.

Notes

Because `txnlog.dat` is a binary file, it can't be read by a normal text editor like sublime or vim. Instead, you'll need to read it programatically and parse the data you read in from there.

This is how the transaction log is structured:

Header:

| 4 byte magic string "MPS7" | 1 byte version | 4 byte (uint32) # of records |

The header contains the canonical information about how the records should be processed. Be sure to validate the magic string from the header to ensure you're parsing the correct file format. Note: there are fewer than 100 records in the sample `txnlog.dat`, this is not true of all transaction logs though.

Record:

| 1 byte record type enum | 4 byte (uint32) Unix timestamp | 8 byte (uint64) user ID |

Record type enum:

- 0x00: Debit
- 0x01: Credit
- 0x02: StartAutopay
- 0x03: EndAutopay

For Debit and Credit record types, there is an additional field, an 8 byte (float64) amount in dollars, at the end of the record.

All multi-byte fields are encoded in network byte order.

The first record in the file, when fully parsed, will have these values:

Record type	**Unix timestamp**	**user ID**	**amount in dollars**
'Debit'	1393108945	4136353673894269217	604.274335557087

Included files

The files for this assessment are included in BackEndAndDevOps.zip