You've been asked by the Product Owner to develop a "smart-banking" feature for our digital banking platform. Your submission to this task will be appraised by our tech team - so this is your opportunity to impress!

Goal

Save customers money by minimising their overdraft fees.

Background

Every customer of the bank has
- a current account with an overdraft facility
- a savings account

Fees are charged by the bank for every day the overdraft facility is used; i.e. when a current account has a negative balance at the end of the day. The fees are proportional to the amount of overdraft used - so the goal is to minimise the debt.

Requirements

We want you to develop a feature that automatically transfers funds from the customer's savings account to their current account to minimise overdraft fees.

To test how your solution performs, the bank will test your application with a series of comma-separated-variable (CSV) files that contains a ledger of transactions. The structure of these files is described in Appendix 1: Customer ledger input file specification

When processing an input CSV file, your solution should output a CSV with the same structure and content of the input file, but with additional entries for the adjustments your system would make to the accounts. For example, if you transfer £10 from the savings account to the current account, 2 new entries would be added to the ledger: the first decrementing the savings account by £10 and the second incrementing the current account

by £10. These adjustments should have a `InitiatorType` value of "`SYSTEM`".

Assumptions and Constraints

All accounts start at £0.

The overdraft is unlimited.

The savings account balance cannot drop below zero.

Unfortunately we will not be able to clarify any requirements on an individual basis, so please ensure you document any assumptions you made when creating your solution.

Out of Scope

You do not need to apply the overdraft fees/interest rates.

We're not looking for a fully-fledged banking application; just some executable code that shows how this feature could work.

A graphical user interface is not required. We expect a simple command line application.

You do not need to persist any data.

Your Submission

You must create a software solution in the programming language of your choice (we use Java and JavaScript, but choose the language you are most comfortable with). Please ensure this language can be easily installed/run on MacOS, Windows or Linux.

We're looking for an elegant and robust solution that is easily understood by a fellow developer. There is no text-book solution!

Please provide the **source code** and detailed instructions on how to download, build and run your software. This should include any prerequisites that must be installed. We recommend using a service such as GitHub to share your code with us.

Deadline & Next Steps

You have 5 days to complete this task.

Once we've received your task submission, we will test it by running it with some sample CSVs that match the structure (but not necessarily the values) of Appendix 2: Example customer ledger input CSV: Filename: "customer-1234567-ledger.csv". We will then review the quality of the source code and any additional materials you provide.

Should we decide to progress your application after reviewing your task… *congratulations!* We will invite you to meet the team remotely. Spaces are limited, so even a functionally-correct submission may not be sufficient to guarantee a place. Show your thinking; over-share your assumptions.

Due to the phenomenal response rate to our graduate scheme, unfortunately we will not be able to provide detailed feedback if your application is rejected.

Good luck!

Appendix 1: Customer ledger input file specification

Comma separated values.

1 file per customer.

File contains ledger of transactions for all of a customer's accounts.

Each row is a transaction.

Assume £0 starting balance.

Legend

- **AccountID**: (Integer) The globally unique ID of the account
- **AccountType**: (String) An enumerated set of account types
- **InitiatorType**: (String) "`ACCOUNT-HOLDER`" if the customer initiated the transaction, or "`SYSTEM`" if we programmatically initiate a transaction.
- **DateTime**: (String) ISO-8601 formatted date-time of the transaction ●

TransactionValue: (Signed floating point number) GBP (£) value of the transaction. Positive values are deposits and will increment account balance. Negative values are withdrawals and will decrement account balance.

Appendix 2: Example customer ledger input CSV: Filename: “customer-1234567-ledger.csv”

```
AccountID,AccountType,InitiatorType,DateTime,TransactionValue
789,CURRENT,ACCOUNT-HOLDER,2018-12-30T09:10:00Z,50.00
123,SAVINGS,ACCOUNT-HOLDER,2018-12-30T10:15:30Z,100.50
123,SAVINGS,ACCOUNT-HOLDER,2018-12-30T22:10:00Z,50.25
789,CURRENT,ACCOUNT-HOLDER,2018-12-30T23:20:00Z,-77.43
789,CURRENT,ACCOUNT-HOLDER,2019-01-01T07:20:00Z,-20.23
123,SAVINGS,ACCOUNT-HOLDER,2019-01-01T09:10:00Z,10.56
789,CURRENT,ACCOUNT-HOLDER,2019-02-01T09:20:00Z,200.23
123,SAVINGS,ACCOUNT-HOLDER,2019-02-03T10:20:15Z,100.23
```