This program is split up into three modules [StatementReport.rkt, AccountTransactions.rkt and AccountInit.rkt]

(StatementReport.rkt is the main program)

In order to run everything, call (generate-statements) in StatementReport.rkt

Example Code Dry Run

STEP 1: The user calls (generate-statements). The out port is created as a new file named STATEMENTS.TXT

STEP 2: All functions in this step can be found within AccountInit.rkt

( file->accounts) is called. This creates the account hash table from the ACCOUNTS.TXT input file.

* The contents of ACCOUNTS.TXT are split into a list of strings representing each line
* Each string is converted into an account through a call of account-input-line->account.
  + The string is converted into an input port, which is used to populate the information in a call to the constructor of the account struct.
* The list of accounts is converted into a hash table of accounts through a call to list->hash. List->hash is passed in a rule to use account-IDs as keys.
  + List->hash recursively calls itself, deconstructing the list and constructing the hash within the parameters of each call. This repeats until the list is empty and the hash contains a key-value pair for each entry in the original list.

STEP 3: The functions for this step can be found in AccountTransactions.rkt

(process-transactions) is called. This iterates through the TRANSACTIONS.TXT input file, appending each transaction to the transaction list within the associated account in the hash table.

* The Account hash and (file->transaction lines) [a list of strings representing transactions] are passed into (process-transactions)
* Process transactions uses the default parameter value for trans ID [10001]
* Process-transactions recursively calls itself to iterate through each entry in the list. Each call raises trans ID by 1 and consumes the next list entry.
  + Each call of process-transactions calls (find-account-and-add-transaction-information). It supplies the transaction-string and the ID
    - The transaction string is converted into an input port, which is used to populate the information in a call to the constructor of the transaction struct.
    - gather-type-specific information is called in order to create either a payment or a purchase to add to the transaction [either the build-purchase or build-payment subfunctions are used depending on which type this function determines for the transaction
    - (new-transaction-on-account) is called, which (functionally) creates a new account with an updated transaction list
    - The customer ID of the transaction Is used to find the relevant account in the hash, which is then (functionially) updated with the new account object

STEP 4: The Functions for this step can be found in StatementReport.rkt

Each account in the hash is iterated through and (print-account-statement) is called for that account

* (statement-header) is called with the current account. It writes the account iD,name and starting balance to output
* (statement-body) is called with the current account. It calls print-transaction which recursively calls itself in order to iterate through each transaction in the account’s transaction list.
  + transaction->string is called, which retrieves the transaction ID, the transaction-type and then the amount
* (statement-footer) is called with the current count. It writes total purchases, payments, and balance to out. Then writes a statement separator to out(a bunch of \* characters)
  + (total-purchases) is called, which calculates the sum amounts from all purchases in the current account. Then formats that as a string that only shows 2 decimal places
  + (total-payments) is called which calculates the sum amounts from all payments in the current account. Then formats that as a string that only shows 2 decimal places
  + (ending-balance) is called which calculates starting balance + purchase-total – payment total on the current account. Then formats that as a string that only shows 2 decimal places

STEP 5: The output port is closed, saving the new STATEMENTS.TXT file