An Extreme Course Project:
QBASIC (Queen's Bank Simulation Console)

CISC 327 - Fall 2017 (Revision 2017.10.02)

Project Teams

You are to form a team of three people to design, implement, document and deliver a two part software product. All phas should follow the eXtreme Programming philosophy as much as it applies - in particular,

- continuously maintained test suites as requirements and quality control
- pair programming of all code
- simplest possible solution to every problem
- continuous redesign and rearchitecting
- automation in testing and integration
- frequent integration and complete releases

Every 1–2 weeks, you will deliver a progress report or other evidence of your team's efforts as required by project assignments.

Project Phases

The project will be done in several phases, each of which will be an assignment. Phases will cover steps in the process of creating a quality software result in the context of an eXtreme Programming process model.

Assignments will be on the quality control aspects of requirements, rapid prototyping, design, coding, integration and analysis of the product you are building. Throughout the project, you should keep records of all evidence of your product quality control steps and evolution, in order to make the marketing case that you have a quality result at the end of the course.

Your final products will be tested and evaluated by an independent evaluator.

Project Schedule

The course project consists of six assignments, with separate handouts for each one. Assignments are scheduled to be due as follows (**preliminary, subject to change**):

Tue Oct 10	Thu Oct 19	Thu Nov 2
Tue Nov 14	Wed Nov 22	Fri Dec 1

Hand-In

Assignments must be handed in as PDF documents in OnQ by 10pm on the due date. Indicate clearly your team and member names on every hand-in. Remember that this is a course in quality - neatness counts, and professional results are expected!

Project Requirements

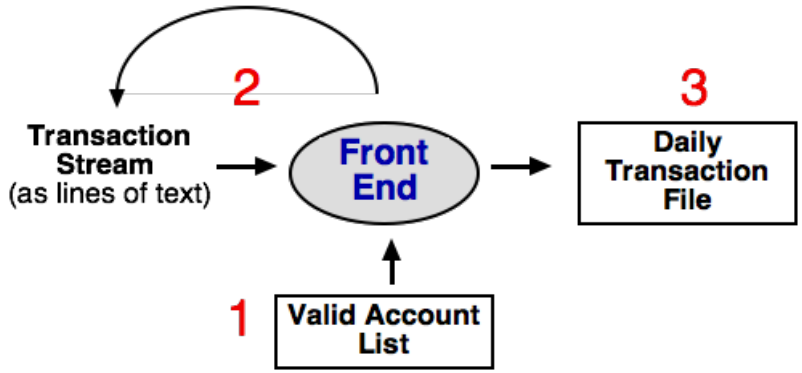
The product you are to design and build is the Queen's Bank Simulation Console ("QBASIC"). QBASIC consists of two parts:

- the Front End, a banking transaction acceptor for simple ATM-style banking transactions
- the Back Office, an overnight batch processor to maintain and update a master accounts file

Both parts will be run as console applications, that is, they are to be invoked from a command line and use text and text file input/output only (this is important for assignments later in the project, so don't ask for exceptions). Avoiding more complex user interfaces in this course allows us to concentrate on other aspects of software quality.

The Front End

The Front End reads in a list of valid account numbers (1), processes a stream of transactions one at a time (2), and writes out a summary file of transactions at the end of the day (3)



Informal Customer Requirements for the Front End

The Front End handles a sequence of transactions, each of which begins with a single transaction code (word of text) on a separate line

The Front End must handle the following transaction codes:

login	- start a Front End session (processing day)
logout	- end a Front End session
createacct	- create a new account (privileged transaction)
deleteacct	- delete an existing account (privileged transaction)
deposit	- deposit to an account (ATM transaction)
withdraw	- withdraw from an account (ATM transaction)
transfer	- transfer between accounts (ATM transaction)

Transaction Code Details

login - start a Front End session (processing day)

- should ask for a type of session, which can be either:
  machine, which means ATM mode, or
  agent, which means privileged (teller) mode
- after the type is accepted, reads in the valid accounts list file (see below) and begins accepting other transactions

Constraints:
- no transaction other than login should be accepted before a login
- no subsequent login should be accepted after a login, until after a logout
- after an ATM login,only ATM transactions and logout are accepted
- after a teller (privileged) login, all transactions are accepted

logout - end a Front End session (processing day)

- should write out the transaction summary file (see below) and stop accepting any transactions except login

Constraints:
- should only be accepted when logged in
- no transaction other than login should be accepted after a logout

createacct - create a new account

- should ask for the new account number and name (as text lines)
- should save these for the transaction summary file, but no transactions on the new account should be accepted

Constraints:
- privileged transaction,only accepted in agent mode
- new account number is exactly seven decimal digits *not* beginning with 0 (e.g., 1234567)
- new account number must be different from all other current account numbers
- new account name is between 3 and 30 alphanumeric characters, possibly including spaces but not beginning or ending with a space

deleteacct - delete an existing account

- should ask for the account number and account name (as text lines)
- should check that the account number is valid, and save the account number and name in the transaction summary file

Constraints:
- only accepted when logged in in agent mode
- no further transactions should be accepted on a deleted account

deposit - deposit to an account

- should ask for the account number and amount to deposit in cents (as text lines)
- should check that the account number is valid, and that the amount is valid

Constraints:
- deposits above $1,000 should be rejected in ATM mode (max $999,999.99 in agent mode)

withdraw - withdraw from an account

- should ask for the account number and amount to withdraw
- should check that the account number is valid, and that the amount is valid

Constraints:
- withdrawals above $1,000 should be rejected in ATM mode (max $999,999.99 in agent mode)
- a total of at most $1,000 can be withdrawn from a single account in a single ATM session (no total limit in agent mode)

transfer - transfer from one account to another

- should ask for the from account number, the to account number and the amount to transfer
- should check that the account numbers are valid, and that the amount is valid

Constraints:
- transfer amounts of more than $1,000 should be rejected in ATM mode (max $999,999.99 in agent mode)

Transaction Summary File

At the end of each session (processing day), when the logout transaction is processed, a transaction summary file is written, listing every transaction made in the session.

Contains transaction messages (text lines) of the form:

```
CCC AAAA MMMM BBBB NNNN
```

where:

`CCC`	is a three-character transaction code, where DEP = deposit, WDR = withdrawal, XFR = transfer, NEW = create account, DEL = delete account, EOS = end of session
`AAAA`	is the first (to) account number
`MMMM`	is the amount in cents (e.g., 123=$1.23)
`BBBB`	is the second (from) account number
`NNNN`	is the account name

Constraints:
- each line is at most 61 characters (plus newline)
- the transaction code is always the first three characters of the message line
- items are separated by exactly one space
- account numbers are always exactly seven decimal digits, not beginning with 0 (e.g., 1000214, 9076522)
- monetary amounts are between 3 and 8 decimal digits, 000 to 99999999, representing $0.00 to $999999.99
- account names are between 3 and 30 alphanumeric characters (A-Z, a-z, 0-9), possibly including spaces, but not beginning or ending with a space (e.g., XYZ, This Acct, My 3rd account)
- unused numeric fields are filled with zeros (e.g., 0000000 for account numbers, 000 for cents)
- unused account name fields are filled with three asterisks: ***
- the set of transactions ends with an end of session (EOS) transaction code

Valid Accounts List File

Consists of text lines containing only an account number.

Constraints:
- each line is exactly 7 characters (plus newline)
- account numbers are always exactly seven decimal digits, not beginning with 0 (e.g., 1000214, 9076522, …)
- the list ends with the special (invalid) account number 0000000

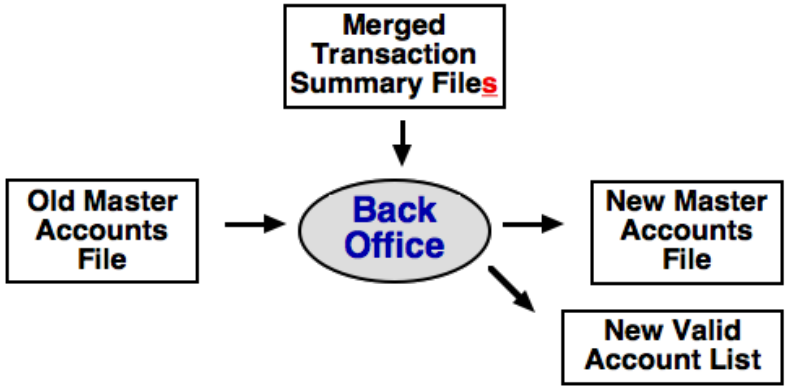
General Requirements for the Front End

The Front End should never crash, and should never stop except as directed by transactions.

The Front End cannot depend on valid input - it must gracefully and politely handle bad input of all kinds (**but** you can assume that the input consists of lines of text).

The Back Office

The Back Office reads in the previous day's master accounts file and then applies all of the transactions in a set of daily transaction files to the accounts to produce today's new master accounts file. Because transactions may also open or close new accounts, it also produces a new valid accounts list for tomorrow's Front End runs.



Informal Customer Requirements for the Back Office

The Back Office reads the Master Accounts File (see requirements) and the Merged Transaction Summary File (see below) and applies all transactions to the master accounts to produce the New Master Accounts File and the Valid Accounts List File.

The Back Office enforces the following business constraints, and produces a failed constraint log on the terminal as it processes transactions.

Constraints:
- no account should ever have a negative balance
- a deleted account must have a zero balance
- a created account must have a new, unused account number
- the name given in a delete transaction must match the name associated with the deleted account

The Master Accounts File

The Master Accounts File consists of text lines of the form:

```
AAAA MMMM NNNN
```

where:

`AAAA`	is the account number
`MMMM`	is the account balance in cents (e.g., 123=$1.23)
`NNNN`	is the account name

Constraints:
- each line is at most 47 characters (plus newline)
- items are separated by exactly one space
- account numbers, monetary amounts, and account names are as described for the transaction summary file above
- the Master Accounts File must always be kept in ascending order by account number

The Merged Transaction Summary File

The Merged Transaction Summary File is the concatenation of any number of Transaction Summary Files output from Front Ends, ending with an empty one—one with no real transactions, just an “end of session” (EOS) transaction code.

The Valid Accounts List File

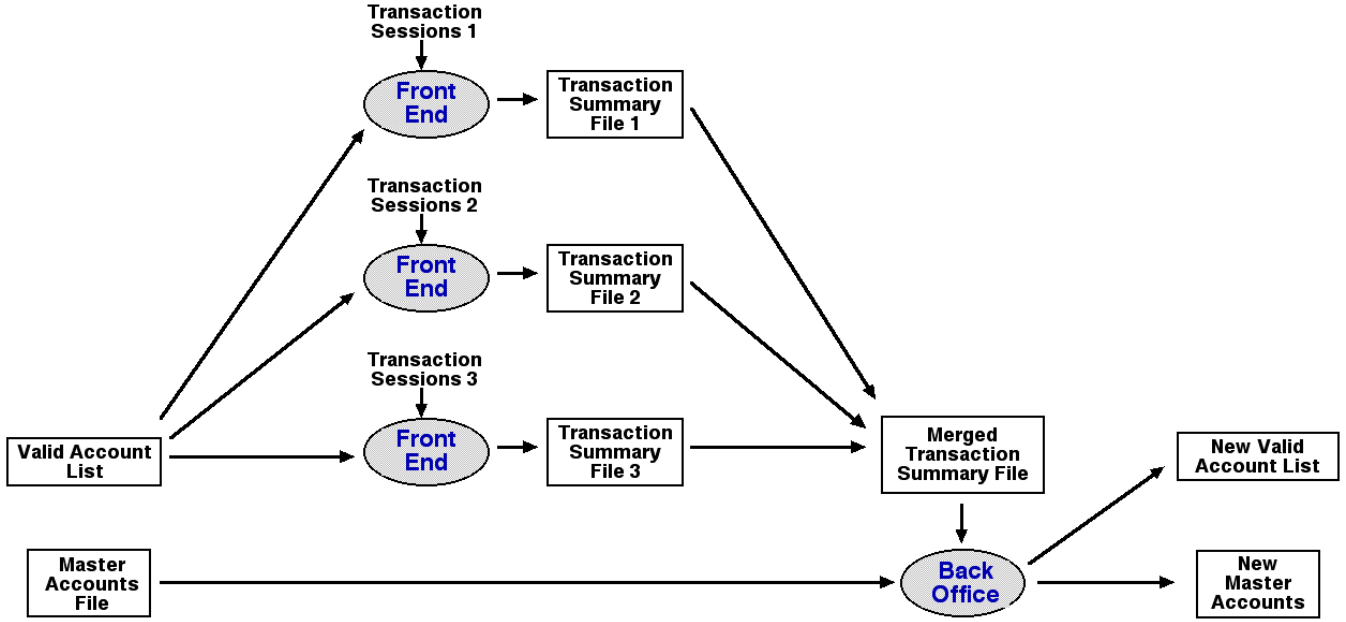
A file containing every active account number in the New Master Accounts File, in the format described for the Front End.

General Requirements for the Back Office

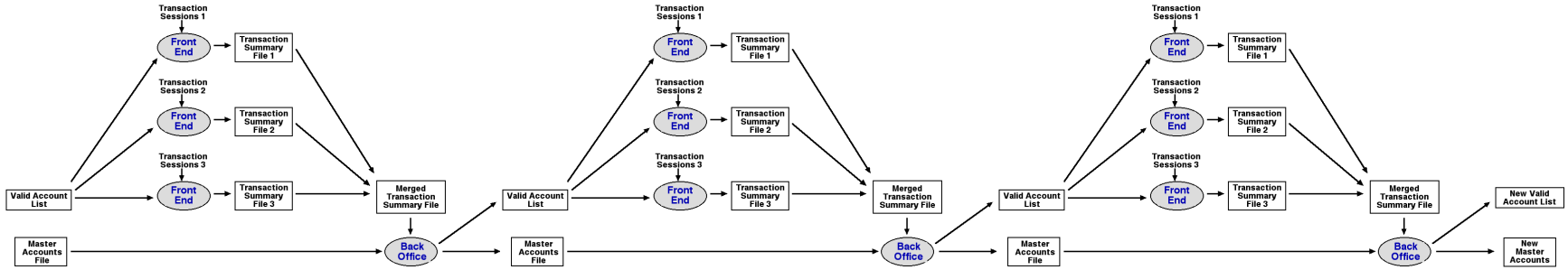
The Back Office uses only internal files, and therefore can assume correct input format on all files. However, values of all fields should be checked for validity, and if the Back End encounters an invalid field, it should immediately stop and log a fatal error on the terminal.

The Integrated QBASIC System

The Front End and Back End fit together to form a complete centralized overnight banking system. Each day several instances of the Front End will run at different locations in the city, independently tracking their day's transactions based on the previous night's valid accounts list from the Back End. In the last assignment of the project, you will integrate your Front End and Back End to form a complete working banking system.



The new valid accounts list and new master accounts file from the end of each day form the valid account list and master accounts file to begin the next day, linking days together into a continuous overnight banking system.



While modern banking systems are much more interactive, this is a simplified version of the actual model used by computerized banking systems for the first three decades of their existence.