An Extreme Course Project:
QIES (Queen's Intercity Excursion System)

CISC / CMPE 327 Fall 2018 (Revision **2018.10.01 [002]**)

Revision 2018.10.01 [002] fixes the cut-off margins in 2018.09.29 [001].

Project Teams

You are to form a team of two to four (preferably *three*) people to design, implement, document and deliver a two-part software product. All phases should follow the eXtreme Programming philosophy as much as it applies:

- continuously maintain test suites as requirements and quality control
- pair-program all code
- choose the simplest possible solution to every problem
- continuously redesign and rearchitect
- automate testing and integration
- frequently integrate complete releases

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

Project Phases

The project has 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.

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**):

(Choose teams: Fri., Sept. 21)

A1: Fri., Oct. 5	A2: Mon., Oct. 15	A3: Wed., Oct. 31
A4: Fri., Nov. 9	A5: Thu., Nov. 22	A6: Fri., Nov. 30

Hand-In

Assignments must be handed in via onQ by 10pm on the due date. Indicate <u>clearly</u> your team and member names on every hand-in. Remember that this is a course in <u>quality</u>; neatness counts, and professional results are expected!

Project Requirements

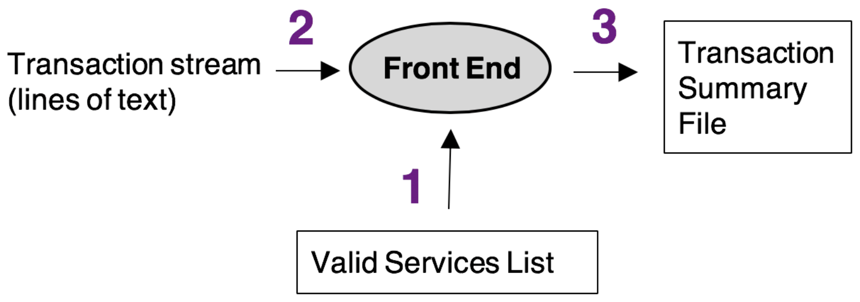
The product you are to design and build is the Queen's Intercity Excursion System ("QIES"). It consists of two parts:

- the Front End, an interface for transactions involving transport tickets
- the Back Office, an overnight batch processor that updates a central manifest file

Both parts will be run <u>as console applications</u>, that is, they are to be invoked from a command line and use text and text file input/output only (this is <u>essential</u> 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 services* (1), processes a stream of *transactions* one at a time (2), and writes out a *transaction summary file* 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:

<u>login</u>	- start a Front End session (processing day)
<u>logout</u>	- end a Front End session
<u>createservice</u>	- create a new service (**planner** transaction)
<u>deleteservice</u>	- delete an existing service (**planner** transaction)
<u>sellticket</u>	- sell a ticket (agent or planner transaction)
<u>cancelticket</u>	- cancel a ticket (agent or planner transaction)
<u>changeticket</u>	- move sold tickets to another service (agent or planner transaction)

Transaction Code Details

login - start a Front End session (processing day)

- should ask for a type of session, which can be either:
  - agent (an ordinary worker), or
  - planner (a worker from the Planning Department)
  - (A planner is "more privileged" than an agent.)
- after the type is accepted, reads in the Valid Services 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 agent login, only agent transactions (including logout) are accepted
- after a planner (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

createservice - create a new service

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

Constraints:
- privileged transaction, only accepted in planner mode
- new service number is exactly five decimal digits *not* beginning with 0 (e.g., 12345)
- new service number must be different from all other current service numbers
- date must be given in the format `YYYYMMDD` where `YYYY` is a 4-digit year, `MM` is a 2-digit month, `DD` is a 2-digit date, and
  - `YYYY` is between 1980 and 2999
  - `MM` is between 1 and 12
  - `DD` is between 1 and 31
- new service name is between 3 and 39 alphanumeric characters, possibly including spaces but not beginning or ending with a space

deleteservice - delete an existing service

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

Constraints:
- only accepted when logged in in planner mode
- no further transactions should be accepted for a deleted service

sellticket - sell tickets for an existing service

- should ask for the service number and number of tickets (as text lines)
- should check that the service number is valid, and that the number of tickets is valid

cancelticket - cancel sold tickets

- should ask for the service number and number of tickets
- should check that the service number is valid, and that the number of tickets is valid

Constraints:
- no more than 10 tickets can be cancelled for a specific service in a single session in "agent" mode (no limit in "planner" mode)
- no more than 20 tickets can be cancelled in a single session in "agent" mode (no limit in "planner" mode)

changeticket - change tickets from one service to another

- should ask for the current service number, the new service number and the number of tickets
- should check that the service numbers are valid

Constraints:
- no more than 20 tickets can be changed in a single session in "agent" mode (no limit in "planner" 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.

This file contains transaction messages (text lines) of the form:

```
CCC AAAA MMMM BBBB NNNNNN YYYYMMDD
```

where:

`CCC`	is a three-character transaction code, where CRE = create service, DEL = delete service, SEL = sell tickets, CAN = cancel tickets, CHG = change tickets, EOS = end of session
`AAAA`	is the first (source) service number
`MMMM`	is the number of tickets
`BBBB`	is the second (destination) service number
`NNNNNN`	is the service name
`YYYYMMDD`	is the service date

Constraints:
- each line is at most 68 characters (plus newline)
- the transaction code is always the first three characters of the message line
- items are separated by exactly one space
- service numbers are always exactly five decimal digits, not beginning with 0 (e.g., 93260, 19450)
- ticket numbers are between 1 and 4 decimal digits, between 1 and 1000
- service names are between 3 and 39 alphanumeric/quote characters (A-Z, a-z, 0-9, `'`), possibly including spaces, but not beginning or ending with a space; examples:

```
KFLA 67
Gananoque
'''Stra'n''g'e 'ButAllowed'
```

- unused numeric fields are filled with zeroes, as follows: 00000 for service numbers, 0 for ticket numbers, **0 for dates**[a]
- unused service name fields are filled with four asterisks: ****
- the set of transactions ends with an end of session (EOS) transaction code

Valid Services List File

Consists of text lines containing only a service number.

Constraints:
- each line is exactly 5 characters (plus newline)
- service numbers are always exactly five decimal digits, not beginning with 0 (e.g., 93260, 19450)
- the list ends with the special (invalid) service number 00000

General Requirements for the Front End

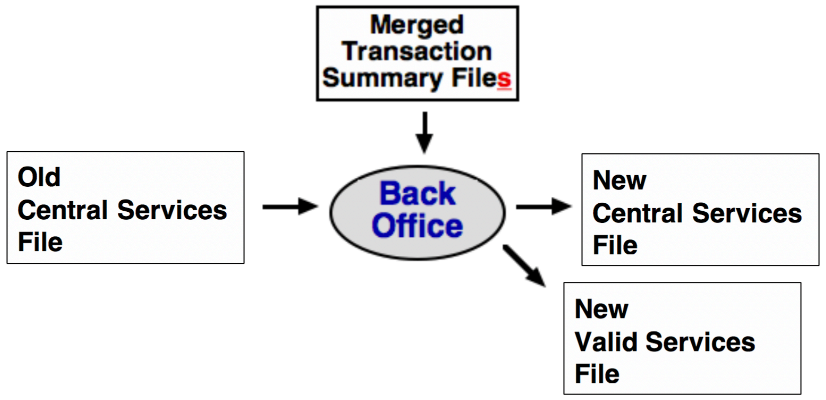
The Front End should never crash, and should never stop except as directed by transactions.

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).

[a] "0 for dates" added 2018–09–29

The Back Office

The Back Office reads in the previous day's Central status file and then applies all of the transactions in a set of daily transaction files to the status file to produce today's new Central status file. Because transactions may create or delete services, it also produces a new Valid Services File for tomorrow's Front End runs.



Merged Transaction Summary Files
Old Central Services File
Back Office
New Central Services File
New Valid Services File

Informal Customer Requirements for the Back Office

The Back Office reads the Central Services File (see requirements) and the Merged Transaction Summary File (see below) and applies all transactions to the Central Services to produce the New Central Services File and the Valid Services 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 service should ever have a capacity of 0 or less
- no service should ever have a capacity greater than 1000
- the number of tickets sold cannot be negative
- the number of tickets sold must not be greater than the capacity
- a created service must have a new, unused service number
- a created service must have zero sold tickets
- a deleted service must have zero sold tickets
- the name given in a delete transaction must match the name associated with the deleted service

The Central Services File

The Central Services File consists of text lines of the form:

```
AAAA CCCC MMMM NNNN
```

where:

`AAAA`	is the service number
`CCCC`	is the service capacity
`MMMM`	is the number of tickets sold
`NNNN`	is the service name
`YYYYMMDD`	is the service date

Constraints:
- each line is at most 63 characters (plus newline)
- items are separated by exactly one space
- service numbers, service names, and dates are as described above for the Transaction Summary File
- service capacities and number of tickets sold are 1 to 4 decimal digits
- service capacity must not be greater than 1000
- the number of tickets sold must not be greater than the service capacity
- the Central Services File must always be kept in ascending order by service 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 Services List File

A file containing every active service number in the New Central Services 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; if the Back Office encounters an invalid field, it should immediately stop and log a fatal error on the terminal.