CISC / CMPE 327 - Fall 2016

Course Project Assignment #2 - Front End Rapid Prototype

Due Thursday October 20

In this assignment, you will design and rapidly program the first version of your team's Front End. Since the next assignment will involve running your Requirements Tests from Assignment #1, **do not do full testing** of your Front End at this time (no marks are for correctness yet!) Try to work together on your program using Pair Programming, where one of you advises on the higher level design issues while the other does the coding.

You should hand in:

(1) A design document for your Front End, giving the overall structure of your solution, showing the classes and methods as a diagram or table, with a brief (one sentence) description of the intention of each one.

(2) The first version of the source code to implement your design. This version should run on at least some inputs, but should **not yet be completely tested** (since that will be the next assignment, and it's better to leave some failures until then).

Your solution to this assignment will be judged on the clarity and readability of the design and the code, so work at using your best programming practices, including naming variables, classes and methods meaningfully and commenting liberally to make it easy for another programmer to understand.

Your solution to this assignment will **not** be judged on its correctness, rather on the quality of its design and coding. This is to be a rapid first version, not a final product. So design and program it to work correctly, but don't worry about getting it fully debugged or tested yet since that's what we'll do next.

Marking Criteria

Assignment 2 will be marked according to the following criteria. In each category, marks will be assigned between zero and the number of marks shown, to a resolution of 1/2 mark.

Design (Primarily in the design document) 4 marks

Architecture
- clearly documents structure of solution
- explicitly describes intention of each class and method
- accurately reflects solution structure
- clearly shows where inputs and outputs fit in

Completeness
- evidently addresses all required functionality
  (solution has parts to address all required operations and results)
- has specified inputs, outputs and files (only!)
  (Front End takes in transactions on std input, produces log on std output, has one input file (valid accounts), and one output file (transaction summary))

Source Code (Details of the source code itself) 4 marks

Structure and format
- code is structured and formatted such that architecture is clearly visible in code
- naming of classes and methods clearly reflect their role in the solution
- as little cloning or redundancy as possible

Maintainability
- avoidance of coding tricks and hacks
- simplest solution possible, no frills and extras
- clearest solution possible, no gratuitous optimizations

Internal documentation
- clear naming of all variables and constants to reflect their role in the solution
- comments for every class and method clearly documenting their interface and intention
- comment at beginning of main program documenting overall program intention, input and output files, and how the program is intended to be run

Overall Presentation & Quality 2 marks

=======

10 marks