Frequently Asked Questions (FAQ)

COMPSCI 280 Assignment 2

Q0. Do we have to use Visual Studio 2012 rather than an earlier version?

A0. Yes. Tutorial 6 provides download instructions. If you use an earlier version of Visual Studio, the code won't look appreciably different, but the .zip of project files won't be compatible for the marker. You'll lose marks if they have to contact you or do the conversion themselves.

Q1. Some of the records in the transaction file have more than one type of error in them. Do we have to report all the different errors in each transaction?

A1. No. You just have to report one type of error that is valid for that transaction in your Part 1 console output and Part 2 validation dialog. You're responsible to get the correct count of transactions with errors for Part 1, but a transaction with multiple types of errors is still just one error transaction.

Q2. You say that the transaction date is in dd/mm/yyyy format, but some of the transaction dates have just a single digit for the day part – do they count as an error?

A2. No. A date such as '2/12/11' should be considered valid.

Q3. You say the Client Number format is "9-digit number; may have leading zeroes" – so does that mean it can have more than 9 digits if the initial ones are zero?

A3. No. The Client Number should be exactly 9 numeric characters, including the check digit. It's just that one or more of the leftmost digits can be a 0, and that's valid. Indeed the only error in the Client Number that you should find is the occasional invalid check digit (i.e. some number other than the one indicated by the Luhn algorithm).

Q4. The Luhn algorithm example at http://en.wikipedia.org/wiki/Luhn_algorithm shows a 10-digit number; will it work for our client IDs which are 9 digits?

A4. Yes, the algorithm works for any length of number. Note that the client IDs in the transaction file are 9 digits *including the check digit* (so it's two digits shorter than the example of Wikipedia). I'd suggest trying the algorithm on paper first (even using shorter numbers, like four digits plus a check) before you start coding it. Also, just as a hint, most of the check digits in the transaction file are correct – if you're finding the majority to be in error, you don't have it coded correctly. Finally,

thanks if class members would refrain from revising the Wikipedia page for the next few weeks – if you got something wrong it could be disruptive to the class.

Q5. The assignment handout says that the transaction file is a list of comma-separated values, but actually it's not: the values are separated by a comma and a space. So do I need to imitate that formatting in my output files?

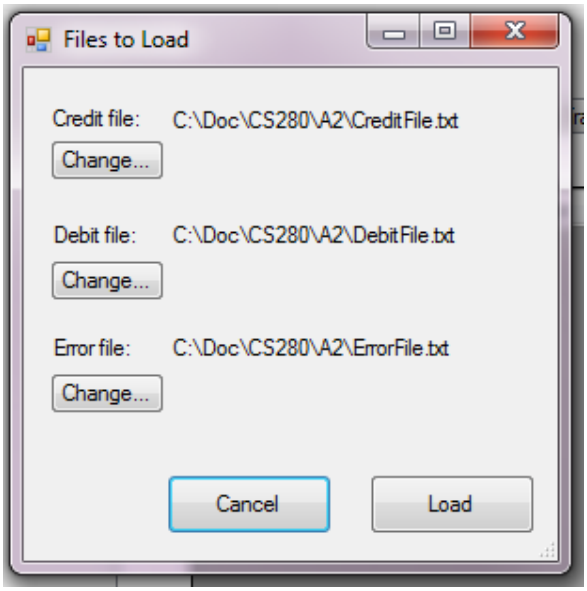
A5. Indeed, while the transaction file is described in the handout as 'separated by commas' the fields are actually separated by a comma and a space. So you need to write your code to recognise the comma separator, but also to deal with the space (note that String variables in C# have a trim() method that will eliminate leading spaces). Moreover, you are required to put those spaces in after the commas when creating the credit, debit and error transaction output files.

Q6. Where do you want to see the 'Correct…' button that pops up a modal dialog for the selected Error transaction? How about double-clicking the row to bring up the same modal window instead?

A6. I had envisioned the 'Correct...' button as being on the tab showing the Error transactions (as this is the only place where it'd apply). Alternatively it could be on a toolbar at the top of the form – but really should be disabled except when the Error transactions are showing… putting the button actually *on* the same tab with the Error transactions seems simpler. I had considered a double-click to invoke the correction dialog (that would be quite natural from a usability standpoint), but didn't want to impose yet another thing for students to figure out by the deadline. So the stated requirement is for a button: use a button.

Q7. For Part 2, you say I have to implement "a modal dialog that allows the user to select the three input files for Credit Transactions, Debit Transactions and Error Transactions, respectively" and that has 'Load' and 'Cancel' buttons. I'm having trouble picturing this. Similar issue for the save dialog. What are you requiring here?

A7. Some things are hard to put into words. In my trial implementation of the assignment (yes, I worked through the whole thing, just to make sure that there weren't any bits that turned out harder than they sounded from the requirements statement) I made a form for the modal dialog that looks like this:



Each 'Change…' button then invokes an OpenFileDialog object to do the actual file picking should you want to change the default. Not saying this is the most beautiful possible solution, but it fits the requirements as stated. I used the same form for the save dialog. That resulted in a fair few if/else statements to implement the Save versus Load logic (including one to use a SaveFileDialog control instead of the OpenFileDialog), but I found that preferable to drawing yet another form and placing all the fiddly label and button controls again.

Q8. Are you going to upload expected output of Error.txt, Credit.txt and Debit.txt so that it is easy for students to check their outputs against what's expected?

A8. Sorry, no. In the real world if you're validating data you don't get a copy of the 'correct' answer – you find what you can find… maybe you get some false-positives (you say it's an error, but it was really OK) and probably false-negatives (you say it's OK, but really there's a problem). I can't stop you from whispering to your friends about how many instances of a bad check digit they found, or such. In fact, I don't mind if you have discussions regarding whether specific records are in error or not. I do ask that you refrain from posting comprehensive logs of errors – that's too much like enabling others to present your work as their own. While substantial assignment points hinge on getting the right error labels and counts, it's not everything; and you can get partial credit for being close. Note that the markers will check for 'hard coded' logic (e.g. if (rec_no==56) Console.WriteLine("Bad Check Digit");) and will give steep deductions.