**cs3307 – Object oriented analysis and design**

**Assignment: Introduction to C++**

[**Banking System Project**](http://www.cppforschool.com/project/banking-system-project.html)

**Introduction:**

This is an individual assignment. You may discuss concepts with colleagues in general terms **but the actual program design, code, and documentation must be your own**.

Please, please, please ... just do not be tempted to copy someone’s program; it just isn’t worth it – not one bit. We will help you if you have any difficulty – come and see us, as many times as \*you\* need to, alright? You see, for any case of plagiarism detected, it then becomes impossible for us to do anything about it other than give a zero. What is the point then? You and we both have to live with this misery for the rest of our lives. Let us not go “there” ... OK, now read on!

You are to write a program (a banking system) in C++. In order to keep focus on learning the language, the system will have text-based menus. By all means, if you want to program a graphics user interface then I will not be a hurdle in your learning. ☺

The following is a description of the approximate system so as to give you an idea of the system. You are to make it more precise by documenting specific requirements the system will implement.

**System description:**

The bank has an unspecified number of customers. For demonstration purposes, you can have a manageable number of customers. The term “client” is a synonym for the term “customer”. Each customer has either or both of a savings account and a chequing account.

There is a bank *manager* who has managerial powers to open and close an account and see the critical details of a particular, or all, customers in a formatted display.

A *customer* has power to use *only* his/her account within the permissible operations. Note that a customer can use multiple operations in one session of system use. When a client wants to open an account, it is assumed that s/he will contact the manager and ask him/her to create the desired account.

Each user of the system, regardless of his/her role type, has an “id” for secure login. After logging into the system, s/he will be bounded by his/her role as to what s/he can do with this system.

The banking system has the following core features (remember this is approximate!):

* Open/Close an account; close restricted to zero balance accounts.
* Deposit to, and withdraw money from, an account.
* Give warning messages for not sufficient funds (e.g., when withdrawing).
* Transfer a sum from one account to another.
* Obtain account balances.
* Give a warning message to the client if the balance on his/her “chequing” account will drop down below the threshold of $1,000 – prior to him/her executing the operation. If s/he decides to go ahead despite the warning message, a charge of $2.00 is levied on the client for each such transaction.
* A client can request the manager to open either savings or chequing account, or both; s/he can request to add the complementary account at anytime.
* The bank manager can display the account details of any given customer, or all customers, and obtain aggregate data on the funds in the bank, etc., categorised appropriately.

There is also a role of a systems *maintenance person* who also has an id to access the system. S/he can turn ON/OFF execution trace. If the trace is ON, then the system will dump the execution traces in a file that can be printed out at a later time by the maintenance person. Each execution trace will log user identification data, time of system access, and the execution trace (which system functions are entered and exited) for the operation conducted by the user. Thus, the system will dump execution traces, one per customer use of the system, in a sequence bounded by ON/OFF trigger.

**IMPORTANT**:

*Minimum set of requirements:* The above set of requirements are a minimum set of requirements for which, if properly implemented, will obtain 70% of this assignment marks.

*Enhancements*: For a further 30% (to make up 100% of this assignment), you are to enhance the minimum set of requirements by another four orthogonal (i.e., distinct or new) requirements of your choice. These requirements should not be minor tweaking of the requirements in the minimum set.

Please note that marks will only be given for enhancements for fully functioning of software corresponding to the requirement in question.

**Project requirements:**

1. **Deliverable 1**: Document textually the minimum set of requirements, categorised by roles, that have been implemented and are demonstratable by the execution of the system. Each requirement should be uniquely identifiable with an “id”.
2. **Deliverable 2**: Identify enhancements requirements in a similar manner.
3. **Deliverable 3**:
   * Deliver C++ source code, along with any instructions on how to compile and run the program and any system configuration parameters.
   * Make the system platform explicit (Windows, Mac, Linux, etc.).
   * Deliver the executable file.
4. **Deliverable 4**:
   * Document five scenarios in **a bullet point** form. Example:
     + Client logs into the ATM.
     + Selects his Deposit account.
     + Checks the balance.
     + Deposits CAD 20.
     + System deposits the funds.
     + System displays the resultant balance.
   * Each scenario must have an appropriate title (e.g., Depositing funds).
   * For each scenario, show using screen dumps, the behaviour of the system as observable from the outside.
   * There should be at least one scenario per role (client, manager, maintenance person).
5. **Deliverable 5**: Describe how the enhanced requirements (see Deliverable 2) have been implemented. In this description, please include names of the relevant software elements (data structures, function names, algorithm, etc.) so as to give a convincing answer.
6. **Deliverable 6**: Please describe what you have learnt about developing the program in C++.
7. All documents should be readable on a Windows machine with standard application programs.
8. **Submission format:** In a zipped folder. The folder name *must* follow the following format:
   * <”Assigment”><underscore ><last name of the owner>
     + E.g., Assignment \_Carpenter
9. **Deadline:** 7th October at 23:59:59.
10. Submitted using the OWL system.

Have fun!