HW5: Bank Accounts - using Dynamic Memory Allocation:

You have been hired as a programmer by a major bank. Your first project is a small banking transaction system. Each account consists of a number and a balance. The user of the program (the teller) can create a new account, as well as perform deposits, withdrawals, balance inquiries, close accounts, etc.**(Note: Some additional features have been added for this assignment.)**

For this assignment, **you must use Classes** and move functionality into the classes. Specifically, you should have at least the following classes:

1. A **Bank** class which consists of an **array of pointers to Accounts** and the **number of accounts** currently active or closed.
2 An **Account** class which consists of a **Depositor, an account number, an account type, account status (open or closed), account balance, an array of Transactions, and the actual number of transactions performed on the account. (Note: creating an account is considered a transaction.)**
3 A **Depositor** class which has a **Name and a social security number**.
4 A **Name** class which consists of **first and last names**.
5. A **Transaction** class which consists of a **transaction type** (e.g., create, deposit, withdrawal, balance, etc.) and a **transaction amount** (where applicable).

As before, you must implement appropriate member functions (or methods) to each class so as to implement the functionality required. New member functions (and general functions) are to be added as necessary.

The data members of each class must be private. As such, you may need to provide accessor and mutator functions.

You must use **separate compilation** of your program modules: **Bank.h, Bank.cpp, Account.h, Account.cpp, Depositor.h, Depositor.cpp, Name.h, Name.cpp, Transaction.h, Transaction.cpp, as well as the module containing the main() function**.

As in previous assignments, initially, the account information of existing customers is to be read into the database. **(Note: you must use dynamic memory allocation as each account is created.)** The bank can handle a maximum of **MAX_NUM** accounts. Each account can have a maximum of **MAX_TRANSACTIONS** transactions. The program keeps tracks of the actual number of currently active accounts. **A table of the initial database of active accounts should be printed.**

As before, the program then allows the user to select from the following menu of transactions:

Select one of the following:
- W - Withdrawal
- D - Deposit
- N - New account **(NOTE: you must use dynamic memory allocation to create a new Account)**
- B - Balance
- I - Account Info (without transaction history)
- H - Account Info plus Account Transaction History
- C - Close Account (close but do not delete the account)
- R - Reopen a closed account
- X - Delete Account (close and delete the account from the database))
- Q - Quit

Once the user has entered a selection, appropriate functions should be called to perform the specific transaction. **At the end, before the user quits, the program prints the contents of the database. You should also print the transaction history for each account in the database.**

As in previous assignments, make sure to use enough test cases so as to completely test program functionality.

EXTRA CREDIT:
1. Create and use a **TransactionInputForm class** for use in **all** transactions as a **screen input form**
2. The TransactionInputForm object must be **dynamically allocated** (and later deleted) for each transaction.
3. Each account, instead of an array of Transactions, has **a pointer to a dynamically allocated array of Transactions**. (Note: this dynamically allocated array of Transactions must be deleted when the account is deleted.)