CSI2372 – Fall 2019
PROJECT

Due Date: By 11:59 PM, Sunday, November 24th, 2019

Evaluation: 26% of final mark (see marking rubric at the end of handout)

Late Submission: none accepted

Purpose: The purpose of this project is to help you learn the whole concepts of C++.

Teams: The assignment must be done in team of 4 students. Team members must be in the same lecture section. Submit one project per team; be sure to have both team members' name in the comments at the top of the project.

General Guidelines When Writing Programs:

- Include the following comments at the top of your source codes

```
// ------------------------------------------------------------------------
// Project (include number)
// Include your name (s) and student id(s))
// For CSI2372 Section (your section) – Fall 2019
// ------------------------------------------------------------------------
```

PROJECT: SIMPLIFIED MANAGEMENT OF BANK ACCOUNTS

Each quarter, the ABC bank must make the state of client accounts it manages. In order to facilitate its task, ABC is speaking to you to produce an account management system. In addition, in order to facilitate the integration of the new system with the bank's exiting applications, the bank requires the system to be implemented in C++.

The existing client accounts information is contained in a text file named **clients.txt** that is structured as follows:

- account Id (integer, length 5),
- account type (integer, 2),
- date of last update (integer, 6),
- balance (double, 8 with 2 digits after the decimal point),
- number of years (integer, 2), for deposits and loans (value of 00 to be ignored for other account types),
- ooan rate (double, 5 with 2 after the decimal point), (00.00 for other types of accounts).
- customer name (alphanumeric).

All customer transactions are saved in a **transact.txt** file whose structure is as follows:

- account Id (integer, 5),
- account type (integer, 2),
- date of the transaction (integer, 6),
- transaction code (integer, 2),
- amount (double, 6 with 2 digits after the decimal point).

The type of account is the different types of accounts that a customer can hold with the bank:

- 01 for a checking account,
- 02 for a savings account,
- 03 for a term deposit,
- 04 for a loan.

The transaction code represents one of the following operations:

- 01 deposit, is done on all types of account (note that a deposit on a loan is in fact a refund),
- 02 withdrawal, is done on the first two types,
- 03 check, only possible on the first type,

In addition:

- A 50 cent ($ 0.50) fee is applied for each withdrawal on a checking or savings account.
- Deposit or refund transactions are free of charge.
- At the balance sheet date, the deposit accounts are enhanced by an annual rate (identical for all accounts).
- To simplify, the total amount to be paid for a loan is calculated from the creation and is equal to the amount read from the **clients.txt** file increases the loan rate over the specified number of years.

On the other hand, the following *transactions* are considered *invalid*:

- Withdrawal of a term deposit or loan.
- Issuing a check for a savings account.
- Withdrawal or check amount greater than the amount available in the account.

REQUIRED WORK

Complete the given program "**BankABC.h**" and "**BankABC.cpp**" by designing and adding them in **.h**, and **cpp.**

Note : The main function is given for the project.

```
//*********************************************************************************************
//* Goal: Sort a list of bank accounts in ascending order of numbers                          *
//* Note: This sort leaves the last count (of number 0) of the list at its position           *
//* to ensure the later tests of the end of the list!!!                                       *
//* Inputs: listAccount (BankAccount *), a list of bank accounts.                             *
//* Outputs: listAccount (BankAccount *), the list of bank accounts sorted.                   *
//********************************************** ***********************************************
void sortAccounts(BankAccount **listAccount);


//*********************************************************************************************
//* Goal: This function reads the file "clients.txt" and builds an array containing           *
//* thedifferent bank accounts of customers.                                                  *
//* Inputs: Nothing                                                                           *
//* Outputs: listAccount (BankAccount *), the list of bank accountsread                       *
//* fromthe file "clients.txt".                                                               *
//*********************************************************************************************
BankAccount ** readAccounts();


//*********************************************************************************************
//* Goal: This function is used to read the file "transact.txt" and to update                 *
//* the accounts concerned by the read transitions.                                           *
//* Entries: listAccount (BankAccount *), the list of bank accounts.                          *
//* Outputs: Nothing.                                                                         *
//*********************************************************************************************
void updateAccounts(BankAccount **listAccount);


//*********************************************************************************************
//* Goal: This function displays the list of bank accounts for all customers.                 *
//* Inputs: listAccount (BankAccount *), the list of bank accounts.                           *
//* Outputs: Nothing                                                                          *
// *********************************************************************************************
void displayAccounts(BankAccount **listAccount);
```

Evaluation Criteria of the project (26 points)

Source code	Marks
Project (26 pts.)	
void trierComptes(CompteBancaire **);	5pts.
CompteBancaire ** lireComptes();	5pts.
void majComptes(CompteBancaire **);	5pts.
void afficherComptes(CompteBancaire **);	5pts.
Displaying the result using variables values	6 pts.