Project CSI2372: In group of 2 Students

PROJECT: **Mini BANK System**

The Mini Bank cooperation must make the state of client accounts. Facilitating its accounting task, the bank is speaking to you to produce an account management system. In addition, to enable integrating 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),
- loan 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 missing parts of the program**.**

Bank.cpp This program reads clients information and transactions from the following 2 files:
- clients.txt
- transact.txt

It also allows certain operations on the read data:
- Display client bank accounts
- Updating client accounts

Bank.h contains the definition of classes and the declaration of the functions.

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 accounts read                 *
//* from the 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 (20 points)

Function	Marks
Project (100 pts.) 20% of final mark	
void sortAccounts (BankAccount **);	25pts.
BankAccount ** readAccounts ();	25pts.
void updateAccounts (BankAccount **);	25pts.
void displayAccounts (BankAccount **);	25pts.
(Bonus) Modularization of the project and designing a build automation using CMake (platform independent build system)	Extra 20 pts.

General Guidelines When Writing Programs:

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

```
// ---------------------------------------------------------------------------
// Your names and student #(s))
// ---------------------------------------------------------------------------
```