Lab 7 – Bank Inheritance – CS-116

1. Define a ***class BankAccount*** which contains the following:
   - Private data member of:
     - double balance
   - Public member functions:
     - Default constructor which will set balance to 0 and accountNum =”NA”
     - Constructor BankAccount(string acct, double a) will set the accountNum to acct and blanace to a
     - deposit (double amount) which will set the balance to amount plus existing balance
     - withdraw (double amount) which will set the balance to existing balance minus the amount. Make sure there is enough fund to be withdraw (that is balance is greater or equal to amount).
     - get_balance() which will return the current balance
     - display_balance() will print out the current account balance
     - set_account_number(string n) will set the accountNum to n
     - get_account_number () will return the current accountNum

   - the class definition must be stored in file ***BankAccount***.h and class implementation must be stored in file ***BankAccount***.cpp

2. Define ***class CheckingAccount*** which will **inherit** ***class BankAccount***, and then add the following to ***class CheckingAccount***:
   - Private data member:
     - Data member: int transactions;
     - Member function: check_for_fee();
       - Increment number of transactions by 1
       - Set the following constant in the function:
         a. const int FREE_TRANSACTIONS = 3;
         b. const int TRANSACTION_FEE = 1;
       - Check if number of transactions is greater than FREE_TRANSACTIONS then deduct the bank balance by the TRANSACTION_FEE
       - Print out the message similar to the one below:

```
cout <<"Number of transactions: " << transactions
   <<" is over free number of transactions of: "
   <<FREE_TRANSACTIONS << endl;
cout <<"Deduct $" << TRANSACTION_FEE << " from account\n";
```

       - Reset transactions to 0
     -
   - Public member function:
     - Default ***CheckingAccount*** constructor which will set transactions to 0

- **CheckingAccount**(string acctNum, double b) **constructor** which will set the accountNum to acctNum and balance to balance (HINT: You can use BankAccount constructor to set these value), and set transactions to 0.
- Member function withdraw (double amount) with do the following:
    - Deduct the balance by calling BankAccount::withdorw(amount)
    - Call member function check_for_fee()
- Member function deposit (double amount) with do the following:
    - deposit the amount by calling BankAccount member function deposit(amount)
    - call member function check_for_fee()
- Member function month_end() which will set number of transactions to 0

- the class definition must be stored in file ***CheckingAccount***.h and class implementation must be stored in file ***CheckingAccount***.cpp. Since ths class will be using class ***BankAccount***, so you have include ***BankAccount.h*** definition header file

3. Define ***class SavingsAccount*** which will **inherit** ***class BankAccount***, and then add the following to ***class SavingsAccount***:
    - Private data member:
        - Data member: double interest_rate;
        - Data member: double min_balance;
    - Public member function:
        - Default ***SavingsAccount*** constructor which will set interest_rate to 0 and min_balance to 0
        - SavingsAccount(string acctNum, double b, double i) constructor will set accountNum to acctNum and balance to b (Use BankAccount). Also set interest_rate to i and min_balance to b.
        - Member function withdraw (double amount) with do the following:
            - Deduct the balance by calling BankAccount::withdorw(amount)
            - Set the min_balance value (if current balance is less than min balance, then set min_blanace to current balance)
        - Member function set_interest_rate(rate) set the monthly interest_rate to rate
        - Member function get_interest_rate( ) will return interest_rate
        - Member function display_balance() will print out:
            - Will print out the account number, interest rate and balance
        - Member function month_end() which do the following:
            - Calculate the interest_earned = min_balance * interest_rate / 100;
            - Deposit interest_earned so your balance will include the interest_earned. You can use BankAccunt member functiond deposit
            - Set the min_balance to the current balance

- the class definition must be stored in file ***SavingsAccount***.h and class implementation must be stored in file ***SavingsAccount***.cpp. Since the class will be using class ***BankAccount***, so you have include ***BankAccount.h*** definition header file

- You MUST use the following test driver program for this lab:

```
#include <iostream>
#include <string>
using namespace std;
/*
 * You need to use this test program for this lab
 *
 * You need to create SavingsAccount.h, CheckingAccout.h and
 * BankAccount.h files.  The c++ codes for these header files
 * are stored in a separate cpp file.
 */

#include "SavingsAccount.h"
#include "CheckingAccount.h"
#include "BankAccount.h"
#include <vector>
#include "printMeFirst.h"

int main ()
{
   printMeFirst("Ron Sha", "CS-116 - Bank Inheritance Lab");
   double amount;

   // Create accounts

   vector < BankAccount > myAccount;
   BankAccount acct1;
   BankAccount acct2 ("S1002", 3000);
   CheckingAccount myChecking;
   SavingsAccount mySaving ("S1001", 2500, 1.25);

   cout << fixed << setprecision (2);
   mySaving.set_account_number ("S1001");
   mySaving.display_balance ();
   acct1.set_account_number ("S1000");
   acct2.set_account_number ("S1002");
   acct1.deposit (2000);
   myAccount.push_back (acct1);
   myAccount.push_back (acct2);
   for (unsigned int n = 0; n < myAccount.size (); n++)
   {
      myAccount[n].display_balance ();
   }

   // withdraw from account acctNum
   string acctNum = "S1002";
   double withdrawAmt = 200;
   for (unsigned int n = 0; n < myAccount.size (); n++)
   {
```

```
#include <iostream>
#include <string>
using namespace std;

/*
 * You need to use this test program for this lab
 *
 * You need to create SavingsAccount.h, CheckingAccout.h and
 * BankAccount.h files.  The c++ codes for these header files
 * are stored in a separate cpp file.
 */

#include "SavingsAccount.h"
#include "CheckingAccount.h"
#include "BankAccount.h"
#include <vector>
#include "printMeFirst.h"

int main ()
{
    printMeFirst("Ron Sha", "CS-116 - Bank Inheritance Lab");
    double amount;

    // Create accounts

    vector < BankAccount > myAccount;
    BankAccount acct1;
    BankAccount acct2 ("S1002", 3000);
    CheckingAccount myChecking;
    SavingsAccount mySaving ("S1001", 2500, 1.25);

    cout << fixed << setprecision (2);
    mySaving.set_account_number ("S1001");
    mySaving.display_balance ();
    acct1.set_account_number ("S1000");
    acct2.set_account_number ("S1002");
    acct1.deposit (2000);
    myAccount.push_back (acct1);
    myAccount.push_back (acct2);
    for (unsigned int n = 0; n < myAccount.size (); n++)
    {
        myAccount[n].display_balance ();
    }

    }

    // withdraw from account acctNum
    string acctNum = "S1002";
    double withdrawAmt = 200;
    for (unsigned int n = 0; n < myAccount.size (); n++)
    {
        if (myAccount[n].get_account_number () == acctNum)
        {
            cout << "Withdraw $" << withdrawAmt
                 <<" from account " << acctNum << endl;
            myAccount[n].withdraw (withdrawAmt);
            myAccount[n].display_balance ();
        }
    }
```

```
    if (myAccount[n].get_account_number () == acctNum)
    {
       cout << "Withdraw $" << withdrawAmt
          <<" from account " << acctNum << endl;
       myAccount[n].withdraw (withdrawAmt);
       myAccount[n].display_balance ();
    }
 }

 myChecking.set_account_number ("C1001");
 amount = 4500;
 cout << "Account: " << myChecking.get_account_number ()
    <<": Deposit " << amount << endl;
 myChecking.deposit (amount);
 myChecking.display_balance ();

 amount = 150;
 cout << "Account: " << myChecking.get_account_number ()
    <<": Deposit " << amount << endl;
 myChecking.deposit (amount);
 myChecking.display_balance ();

 amount = 100;
 cout << "Account: " << myChecking.get_account_number ()
    <<": Withdraw " << amount << endl;
 myChecking.withdraw (amount);
 myChecking.display_balance ();

 amount = 550;
 cout << "Account: " << myChecking.get_account_number ()
    <<": Withdraw " << amount << endl;

 myChecking.withdraw (amount);
 myChecking.display_balance ();

 cout << "\nSaving account month end\n";
 mySaving.display_balance ();
 mySaving.month_end ();
 mySaving.display_balance ();

 amount = 4000;
 cout << "Account: " << mySaving.get_account_number ()
    <<": Withdraw " << amount << endl;
 mySaving.withdraw (amount);
 mySaving.display_balance ();

 return 0;
}
```

```
58
59      myChecking.set_account_number ("C1001");
60      amount = 4500;
61      cout << "Account: " << myChecking.get_account_number ()
62           <<": Deposit " << amount << endl;
63      myChecking.deposit (amount);
64      myChecking.display_balance ();
65
66      amount = 150;
67      cout << "Account: " << myChecking.get_account_number ()
68           <<": Deposit " << amount << endl;
69      myChecking.deposit (amount);
70      myChecking.display_balance ();
71
72      amount = 100;
73      cout << "Account: " << myChecking.get_account_number ()
74           <<": Withdraw " << amount << endl;
75      myChecking.withdraw (amount);
76      myChecking.display_balance ();
77
78      amount = 550;
79      cout << "Account: " << myChecking.get_account_number ()
80           <<": Withdraw " << amount << endl;
81
82      myChecking.withdraw (amount);
83      myChecking.display_balance ();
84
85      cout << "\nSaving account month end\n";
86      mySaving.display_balance ();
87      mySaving.month_end ();
88      mySaving.display_balance ();
89
90      amount = 4000;
91      cout << "Account: " << mySaving.get_account_number ()
92           <<": Withdraw " << amount << endl;
93      mySaving.withdraw (amount);
94      mySaving.display_balance ();
95
96      return 0;
97  }
```

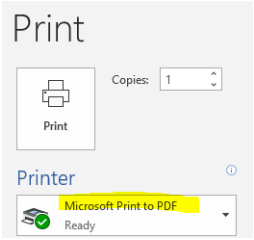
The output of the program is similar to below:

```
Program written by: Ron Sha
Course Info: CS-116 - Bank Inheritance Lab
Date: Tue Nov 03 12:47:11 2020


Saving Account S1001: interest rate: 1.25
Account: S1001 balance is: 2500.00

Account: S1000 balance is: 2000.00

Account: S1002 balance is: 3000.00
Withdraw $200.00 from account S1002

Account: S1002 balance is: 2800.00
Account: C1001: Deposit 4500.00
Checking Account balance:

Account: C1001 balance is: 4500.00
Number of Checking Account transactions: 1
Account: C1001: Deposit 150.00
Checking Account balance:

Account: C1001 balance is: 4650.00
Number of Checking Account transactions: 2
Account: C1001: Withdraw 100.00
Checking Account balance:

Account: C1001 balance is: 4550.00
Number of Checking Account transactions: 3
Account: C1001: Withdraw 550.00
Number of transactions: 4 is over free number of transactions of: 3
Deduct $1 from account
Checking Account balance:

Account: C1001 balance is: 3999.00
Number of Checking Account transactions: 0

Saving account month end

Saving Account S1001: interest rate: 1.25
Account: S1001 balance is: 2500.00

Saving Account S1001: interest rate: 1.25
Account: S1001 balance is: 2531.25
Account: S1001: Withdraw 4000.00
***Not enough fund in account S1001:  Current balane: 2531.25 Withdraw: 4000.00
Saving Account S1001: interest rate: 1.25
Account: S1001 balance is: 2531.25


------------------
(program exited with code: 0)
```

Lab Submission: See the lab submission requirements published in canvas.

To submit your assignment in canvas, you must submit TWO files (one pdf and one zip) as follows:

1. **Attach pdf file** which contains source codes you have written and program output screenshot so I can easily read in one file. You can use a word editor to place all the required programs and screenshots, and then use 'Print' to 'Microsoft Print to PDF' to save to a pdf file



The pdf file MUST have the following sections (1. Program Description, 2. Program Source Code and 3. Program Output).

1. Program Description

- brief *description* of the purpose of the *program and*
- an explanation of what your software does and what problem it solves

2. Program Source Code

- Include all the source codes (program files) you have written for this lab (screenshots are ok)
- Your program must have adequate documentation for your source codes:
  - Program description – see above on Program Description
  - **You must put the following function headers for each function (the function header MUST be placed just above the function declaration in your source code). For Functions, it must have:**
    - *Function name: name of this function*
    - *Function description: the purpose of this function and how to use it*
    - *@param param_name and what the parameter/argument is*

used for

- *@return what is returned from the function*

- Make sure your screenshots are readable (not too small)

Below is an example of source code

File: print_me_first_main.cpp (list the program file individually)

```cpp
//print_me_first_main.cpp
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

/**
 * @Purpose - this function print out the person who wrote the program,
 * and date/time the program run.
 * @parm - name - the author of the program
 * @parm - courseInfo - the name of the course
 * @return - none
 * @author - Ron Sha
 **/

void PrintMeFirst(string name, string courseInfo)
{
    cout <<" Program written by: "<< name << endl; // put your name here
    cout <<" Course info: "<< courseInfo << endl;
    time_t now = time(0); // current date/time based on current system
    char* dt = ctime(&now); // convert now to string for
    cout << " Date: " << dt << endl;
}
```

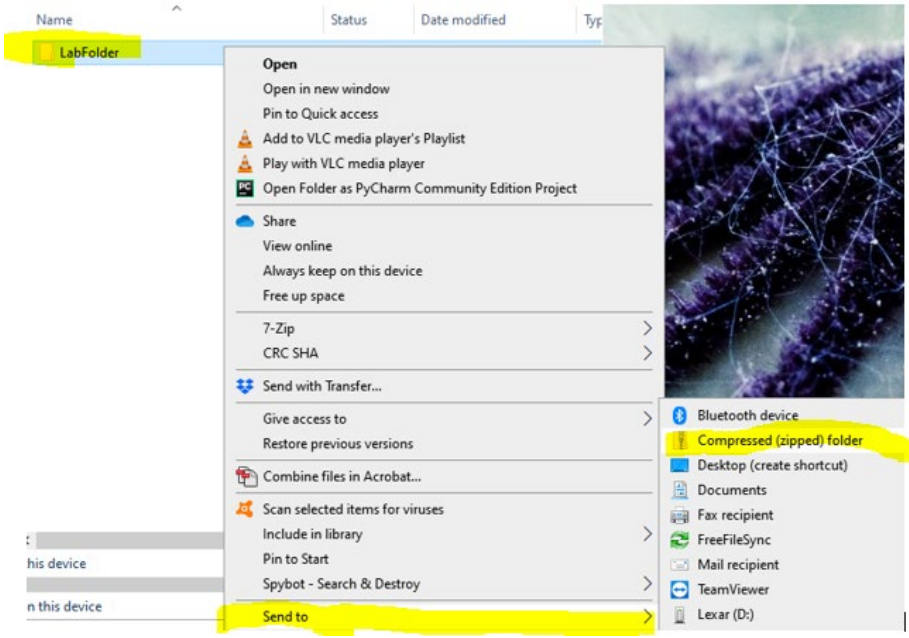
File: list other cpp files separately

3. Program Output

- Attach all the program outputs (screenshots)
- Don't place Source code and program output side by side as it is not readable in screenshot
- Make sure your screenshots are readable (not too small)
- Your main program outputs MUST include your name printout (use the print_me_first function/program).

2. **Attach zip file** which contains all your source code (you can zip the folder) and functions. Even if you only have one source file, you MUST still do a zip file of the folder. I must be able to compile and run your program from all the source code programs after I unzip your zip file.

   You should create a folder for each lab, and place all your programs, functions and all other files related to this lab in this lab folder. To submit the lab folder, you can use "Send to -> compress" in window file explorer to create a zip file of the folder.



3. **Now you can upload both the pdf file and zip file separately as your assignment submission in canvas.**