CS 201, Spring 2017
Homework Assignment 3

Due: 23:59, May 16, 2017

In this homework, you will implement a banking system using linked lists. The bank has multiple customers where each customer has an id, first name and last name. The customers can have multiple accounts where each account has an id and a balance.

In your implementation, you **MUST** keep the customers in a sorted linked list where the customers are stored as sorted according to their last names. For each of these customers, you **MUST** use another sorted linked list to store her/his accounts where the accounts are stored as sorted according to account ids.

The banking system will have the following functionalities; the details of these functionalities are given below:

1. Add a customer
2. Delete a customer
3. Add an account for a customer
4. Delete an account
5. Show the list of all accounts
6. Show the list of all customers
7. Show detailed information about a particular customer

Add a customer: The banking system will allow to add a new customer indicating her/his customer id, first name and last name. Since the customer ids are unique, the system should check whether or not the specified customer id already exists (i.e., whether or not it is the id of another customer), and if the customer exists, it should not allow the operation and display a warning message. Use the following conditions for sorting the customers in the list:

- The customers should be sorted according to last names.
- If there are more than one customer with the same last name, you should compare the first names.
- If the banking system contains a customer with the same first name and last name as the new input, you should not add this customer again and should display a warning message.

Delete a customer: The banking system will allow to delete an existing customer indicating her/his customer id. If the customer does not exist (i.e., if there is no customer with the specified id), the system should display a warning message. Note that this operation will also delete all accounts for the customer of interest.

Add an account for a customer: The banking system will allow to add a new account for a particular customer. For that, the customer id and the deposited amount have to be specified. The system should check whether or not this customer exists; if she/he does not, it should prevent to add an account and display a warning message. If the customer exists, a unique account id is generated and a new account is created with the specified deposit amount. The system should return this account id to the user. The accounts are stored as sorted in ascending order of account ids. If the customer does not exist, the returned account id is -1. Note that the account ids are unique within the banking system; thus, by using an account id, the user can access this account.

Delete an account: The banking system will allow to delete an existing account indicating its account id. If the account does not exist (i.e., if there is no account with the specified id), the system should display a warning message.

Show the list of all accounts: The banking system will allow displaying a list of all the accounts. This list includes the account id, the customer id, the customer name, and the account balance. Note that the accounts have to be shown in sorted order with respect to the last names of the customers they belong to. If there are more than one customer with the same last name, those should be shown in sorted order with respect to their first names.

Show the list of all customers: The banking system will allow displaying a list of all the customers. This list includes the customer id and the customer name. Note that the customers have to be shown in sorted order with respect to the last names. If there are more than one customer with the same last name, those should be shown in sorted order with respect to their first names.

Show detailed information about a particular customer: The banking system will allow to specify a customer id and display detailed information about that particular customer. This information includes the customer id, the customer name, the number of accounts opened by this customer, the list of accounts owned by this customer including the account id and the account balance for each account.

If the customer does not exist (i.e., if there is no customer with the specified customer id), the system should display a warning message.

Below is the required `public` part of the `BankingSystem` class that you must write in this assignment. The name of the class must be `BankingSystem`, and must include these public member functions. We will use these functions to test your code. The interface for the class must be written in a file called `BankingSystem.h` and its implementation must be written in a file called `BankingSystem.cpp`. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution.

```
class BankingSystem {

public:
    BankingSystem();
    ~BankingSystem();

    void addCustomer( const int customerId, const string firstName, const string lastName );
    void deleteCustomer( const int customerId );

    int addAccount( const int customerId, const double amount );
    void deleteAccount( const int accountId );

    void showAllAccounts();
    void showAllCustomers();
    void showCustomer( const int customerId );
};
```

Here is an example test program that uses this class and the corresponding output. We will use a similar program to test your solution so make sure that the name of the class is `BankingSystem`, its interface is in the file called `BankingSystem.h`, and the required functions are defined as shown above.

Example test code:

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

#include "BankingSystem.h"

int main() {

    BankingSystem B;

    B.addCustomer( 1234, "Cigdem", "Gunduz Demir" );
    B.addCustomer( 4567, "Ercument", "Cicek" );
    B.addCustomer( 891234, "Serhan", "Yilmaz" );
    B.addCustomer( 891234, "Can Fahrettin", "Koyuncu" );
    B.addCustomer( 5678, "Necmi", "Acarsoy" );
    B.addCustomer( 8901, "Cigdem", "Gunduz" );
    B.addCustomer( 9876, "Cigdem", "Gunduz" );

    B.deleteCustomer( 5678 );
    B.deleteCustomer( 1267 );

    int X, Y, Z, Q;
    B.addAccount( 4567, 100.00 );
    X = B.addAccount( 1234, 200.00 );
    Y = B.addAccount( 4567, 300.00 );
    B.addAccount( 4567, 1000.00 );
    Z = B.addAccount( 8901, 100.00 );
    B.addAccount( 5678, 100.00 );
    B.addAccount( 1234, 500.00 );

    B.deleteAccount( X );
    B.deleteAccount( Y );
    B.deleteAccount( Z );

    B.addAccount( 891234, 500.00 );
    B.addAccount( 4567, 200.00 );
    B.addAccount( 1234, 1500.00 );
    Q = B.addAccount( 1234, 300.00 );
    B.deleteAccount( Q );

    B.showAllAccounts();
    B.showAllCustomers();
    B.showCustomer( 8901 );
    B.showCustomer( 4567 );
    B.showCustomer( 1212 );

    return 0;
}
```

Output of the example test code:

```
Customer 1234 has been added
Customer 4567 has been added
Customer 891234 has been added
Customer 891234 already exists
Customer 5678 has been added
Customer 8901 has been added
Customer Cigdem Gunduz already exists
Customer 5678 has been deleted
Customer 1267 does not exist

Account 1 added for customer 4567
Account 2 added for customer 1234
Account 3 added for customer 4567
Account 4 added for customer 4567
Account 5 added for customer 8901
Customer 5678 does not exist
Account 6 added for customer 1234
Account 2 has been deleted
Account 3 has been deleted
Account 5 has been deleted
Account 7 added for customer 891234
Account 8 added for customer 4567
Account 9 added for customer 1234
Account 10 added for customer 1234
Account 10 has been deleted

Account id  Customer id     Customer name           Balance
1           4567            Ercument Cicek          100.00
4           4567            Ercument Cicek          1000.00
8           4567            Ercument Cicek          200.00
6           1234            Cigdem Gunduz Demir     500.00
9           1234            Cigdem Gunduz Demir     1500.00
7           891234          Serhan Yilmaz           500.00

Customer id     Customer name
4567            Ercument Cicek
8901            Cigdem Gunduz
1234            Cigdem Gunduz Demir
891234          Serhan Yilmaz

Customer id: 8901   Customer name: Cigdem Gunduz     Number of accounts: 0

Customer id: 4567   Customer name: Ercument Cicek    Number of accounts: 3
Accounts of this customer:
Account id  Balance
1           100.00
4           1000.00
8           200.00

Customer 1212 does not exist
```

Notes:

1. This assignment is due by 23:59, May 16th, 2017. This homework will be graded by your TA Serhan Yilmaz (serhan.yilmaz@bilkent.edu.tr) and you are supposed to e-mail the homework to him with subject line: CS 201 - HW3. As he is in charge of this homework, you can ask your questions directly to him. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

2. You must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. We will test your implementation by writing our own driver `.cpp` file which will include your header file. For this reason, your class' name MUST BE "`BankingSystem`" and your files' name MUST BE "`BankingSystem.h`" and "`BankingSystem.cpp`". We also recommend you to write your own driver file to test each of your functions. However, you MUST NOT submit this test code (we will use our own test code). In other words, your submitted code for the should not include any `main` function.

3. You must use your own implementation of linked lists. In other words, you cannot use any existing linked list code from other sources such as the `list` class in the C++ standard template library (STL).

   Additionally, you will get NO points if you use an array-based implementation of the list or if you use any other data structures such as vector/array from STL.

   However, if necessary, you may define additional data members and member functions. You may also define additional classes.

   You are not allowed to use any global variables or any global functions.

4. Your code must not have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct.

5. Your implementation should consider all names as case sensitive. For example, last names Ozsoyoglu and OZSOYOGLU should be considered as different names. Also Özsoyoğlu are Ozsoyoglu different last names due to Turkish characters.

6. You are free to write your programs in any environment (you may use either Linux or Windows). On the other hand, we will test your programs on "`dijkstra.ug.bcc.bilkent.edu.tr`" and we will expect your programs to compile and run on the "`dijkstra`" machine. If we could not get your program properly work on the "`dijkstra`" machine, you would lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on "`dijkstra.ug.bcc.bilkent.edu.tr`" before submitting your assignment.