CS 201, Spring 2023
Homework Assignment 1

Due: 23:59, April 7, 2023

In this homework, you will implement a banking system. The bank has multiple branches and multiple customers. The customers can have multiple accounts in different branches.

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

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

Add a branch: The banking system will allow to add a new branch indicating its branch id and branch name. Since the branch ids are unique, the system should check whether or not the specified branch id already exists (i.e., whether or not it is the id of another branch), and if the branch exists, it should not allow the operation and display a warning message.

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

Add a customer: The banking system will allow to add a new customer indicating her/his customer id and 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.

Delete a customer: The banking system will allow to delete an existing customer indicating its 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 in a branch: The banking system will allow to add a new account for a particular customer in a particular branch. For that, the branch id, the customer id, and the deposited amount have to be specified. The system should first check whether or not this branch exists; if it does not, it should prevent to add an account and display a warning message. The system should also check whether or not this customer exists; if it does not, it should prevent to add an account and display a warning message. If both the branch and the customer exist, a unique account id is generated and a new account is created with the speficied deposit amount. The system should return this account id to the user. If the branch or the customer do 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. Note also that the same customer can have multiple accounts at the same branch but their account ids must be different.

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 to display a list of all the accounts. This list includes the account id, the branch id, the branch name, the customer id, the customer name, and the account balance.

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

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

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, the branch id, the branch name, 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 addBranch( const int branchId, const string branchName );
    void deleteBranch( const int branchId );

    void addCustomer( const int customerId, const string customerName );
    void deleteCustomer( const int customerId );

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

    void showAllAccounts();
    void showBranch( const int branchId );
    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.addBranch( 1451, "Bilkent" );
    B.addBranch( 2435, "Kizilay" );
    B.addBranch( 1672, "Bahcelievler" );
    B.addBranch( 3216, "Ulus" );
    B.addBranch( 2435, "Kizilay" );
    B.deleteBranch( 1672 );
    B.deleteBranch( 1723 );
    B.addBranch( 9876, "Umitkoy" );

    B.addCustomer( 1234, "Ali Ak" );
    B.addCustomer( 4567, "Aynur Dayanik" );
    B.addCustomer( 891234, "Cihan Erkan" );
    B.addCustomer( 891234, "Hakan Karaca" );
    B.addCustomer( 5678, "Mustafa Gundogan" );
    B.addCustomer( 8901, "Can Kara" );
    B.deleteCustomer( 5678 );
    B.deleteCustomer( 1267 );

    int account1 = B.addAccount( 1451, 4567, 100.00 );
    int account2 = B.addAccount( 1451, 1234, 200.00 );
    int account3 = B.addAccount( 3216, 4567, 300.00 );
    int account4 = B.addAccount( 1451, 4567, 1000.00 );
    int account5 = B.addAccount( 1672, 8901, 100.00 );
    int account6 = B.addAccount( 2435, 5678, 100.00 );
    int account7 = B.addAccount( 3216, 1234, 500.00 );
    B.deleteAccount( account2 );
    B.deleteAccount( account5 );
    B.deleteAccount( account7 );
    int account8 = B.addAccount( 2435, 891234, 500.00 );

    B.showAllAccounts();
    B.showBranch( 1451 );
    B.showBranch( 1672 );
    B.showBranch( 9876 );
    B.showCustomer( 1234 );
    B.showCustomer( 4567 );
    B.showCustomer( 1212 );
    B.deleteBranch( 1451 );
    B.showCustomer( 4567 );
    return 0;
}
```

Output of the example test code:

```
Branch 1451 has been added
Branch 2435 has been added
Branch 1672 has been added
Branch 3216 has been added
Branch 2435 already exists
Branch 1672 has been deleted
Branch 1723 does not exist
Branch 9876 has been added

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 5678 has been deleted
Customer 1267 does not exist

Account 1 added for customer 4567 at branch 1451
Account 2 added for customer 1234 at branch 1451
Account 3 added for customer 4567 at branch 3216
Account 4 added for customer 4567 at branch 1451
Branch 1672 does not exist
Customer 5678 does not exist
Account 5 added for customer 1234 at branch 3216
Account 2 has been deleted
Account -1 does not exist
Account 5 has been deleted
Account 6 added for customer 891234 at branch 2435

Account id  Branch id   Branch name     Customer id     Customer name             Balance
1           1451        Bilkent         4567            Aynur Dayanik             100.00
3           3216        Ulus            4567            Aynur Dayanik             300.00
4           1451        Bilkent         4567            Aynur Dayanik             1000.00
6           2345        Kizilay         891234          Cihan Erkan               500.00

Branch id: 1451   Branch name: Bilkent     Number of accounts: 2
Accounts at this branch:
Account id  Customer id     Customer name         Balance
1           4567            Aynur Dayanik         100.00
4           4567            Aynur Dayanik         1000.00

Branch 1672 does not exist

Branch id: 9876   Branch name: Umitkoy     Number of accounts: 0

Customer id: 1234   Customer name: Ali Ak    Number of accounts: 0

Customer id: 4567   Customer name: Aynur Dayanik      Number of accounts: 3
Accounts of this customer:
Account id  Branch id       Branch name         Balance
1           1451            Bilkent             100.00
3           3216            Ulus                300.00
4           1451            Bilkent             1000.00
```

```
Customer 1212 does not exist

Branch 1451 has been deleted

Customer id: 4567   Customer name: Aynur Dayanik      Number of accounts: 1
Accounts of this customer:
Account id  Branch id       Branch name         Balance
3           3216            Ulus                300.00
```

IMPORTANT NOTES:
Do not start your homework before reading these notes!!!

NOTES ABOUT IMPLEMENTATION:

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use dynamically allocated arrays in your implementation. You will get no points if you use fixed-sized arrays, linked lists or any other data structures such as vectors/arrays from the standard library. However, if necessary, you may define additional data members and member functions.
2. Moreover, you ARE NOT ALLOWED to use any global variables or any global functions.
3. Output message for each operation MUST match the format shown in the output of the example code.
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. To detect memory leaks, you may want to use Valgrind which is available at `http://valgrind.org`.
5. Otherwise stated in the description, you may assume that the inputs for the functions are always valid (e.g., amount to add an account is a valid positive integer number) so that you do not need to make any input checks.

NOTES ABOUT SUBMISSION:

1. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. Your class name MUST BE `BankingSystem` and your file names MUST BE `BankingSystem.h` and `BankingSystem.cpp`. Note that you may write additional class(es) in your solution.
2. The code (`main` function) given above is just an example. We will test your implementation using different scenarios, which will contain different function calls. Thus, do not test your implementation only by using this example code. We recommend you to write your own driver files to make extra tests. However, you MUST NOT submit these test codes (we will use our own test code). In other words, do not submit a file that contains a function called `main`.
3. You should put all of your `.h` and `.cpp` files into a folder and zip the folder (in this zip file, there should not be any file containing a `main` function). The name of this zip file should conform to the following name convention: secX-Firstname-Lastname-StudentID.zip where X is your section number. The submissions that do not obey these rules will not be graded.
4. Make sure that each file that you submit (each and every file in the archive) contains your name, section, and student number at the top as comments.

5. You are free to write your programs in any environment (you may use Linux, Windows, MacOS, etc.). 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. Thus, we recommend you to make sure that your program compiles and properly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment.

6. This assignment is due by 23:59 on Friday, April 7, 2023. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy.

7. We use an automated tool on dijkstra as well as manual inspection to grade your submissions. Please make sure you submissions will pass the sample test cases successfully. Please note that additional new test cases will also be used for grading purposes.

8. We use an automated tool as well as manual inspection to check your submissions against cheating. Please see the course home page for further discussion of academic integrity and the honor code for programming courses in our department.

9. This homework will be graded by your TA Cihan Erkan (cihan.erkan at bilkent.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on Moodle for questions.