CS 201, Summer 2016
Homework Assignment 3

Due: 23:59, July 20, 2016

In this homework assignment, you will implement a phone-book by using linked lists. In this phone-book, for each person, you will have an entry in which you keep the first name and the last name of the person as well as a list of her/his phone numbers. Each phone number is represented by a three digit integer corresponding to the area code and a seven digit integer corresponding to the number. The phone-book **MUST** use a sorted linear doubly linked list (with no dummy head node) to store the person entries, and for each person, an unsorted circular singly linked list (with no dummy head node) to store the phone numbers.

Below is the required `public` part of the phone book class that you must write in this assignment. The name of the class must be `PhoneBook`, 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 `PhoneBook.h` and its implementation must be written in a file called `PhoneBook.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 PhoneBook {

public:
    PhoneBook();
    ~PhoneBook();

    void addPerson( const string firstName, const string lastName );
    void deletePerson( const string firstName, const string lastName );

    void addPhoneNumber( const string firstName, const string lastName,
                         const int areaCode, const int number );
    void deletePhoneNumber( const string firstName, const string lastName,
                            const int areaCode, const int number );

    void showPerson( const string firstName, const string lastName );
    void showPhoneNumber( const int areaCode, const int number );
    void showAll();

};
```

The details of these functions are given below:

Add a person: This function adds an entry to the phone-book for a given person whose first name and last name are specified as parameters. Note that this function only adds the name to the phone-book without adding any phone numbers. In this function, you should consider the following issues:

- The phone-book should be alphabetically sorted according to last names,
- If you have more than one entry with the same last name, you should also consider first names,
- If your list contains an entry with the same first name and last name, you should not add this person again in your phone-book and should display a warning message to the user.

Delete a person: This function removes the entry from the phone-book for a given person whose first name and last name are specified as parameters. If the phone-book does not have an entry with the specified name, you should display a warning message.

Add a phone number: This function adds a phone number (area code + number) to the entry for which the first and last names are specified. If the phone-book does not have an entry with the specified name, you should display a warning message. Similarly, if the entry exists and contains the same phone number, you should also display a warning message.

Delete a phone number: This function removes a phone number (area code + number) from the entry for which the first and last names are specified. If the phone-book does not have an entry with the specified name, you should display a warning message. Similarly, if the entry exists but does not have that phone number, you should also display a warning message.

Show detailed information about a particular person: This function displays detailed information for the person for which the first and last names are specified. If the phone-book does not have an entry with the specified name, you should display a warning message. The following information should be displayed on the screen:

```
First-name Last-name
Phone number 1
Phone number 2
. . .
```

Show detailed information about a particular phone number: This function displays detailed information for the phone number for which the area code and number are specified. If the phone-book does not have any entry that contains the specified number, you should display a warning message. The following information should be displayed on the screen:

```
First-name Last-name
Phone number
```

Show the list of all entries: This function displays detailed information about all entries in the phone book. An example output can be as follows:

```
Selim Aksoy
312-2903405
312-1234567

Ercument Cicek
312-2906941
532-9876543

Cigdem Gunduz-Demir

Altay Guvenir
312-2901252
312-2901218
```

Important Notes:

1. This assignment is due 23:59 on Wednesday, July 20, 2016. You should upload your homework using the online submission form on the course web page before the deadline. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy as well as academic integrity.

2. For this assignment, 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). However, you can adapt the linked list codes in the Carrano book. You will get no points if you do not use linked lists as indicated. You will also get no points if you implement array-based solutions (using fixed-sized arrays, dynamically allocated arrays, data structures such as vector from the standard library, etc.).

3. Remember that the person entries should be stored in a sorted linear doubly linked list (with no dummy head node) and phone numbers in each person entry should be stored in an unsorted circular singly linked list (with no dummy head node).

4. You are not allowed to modify the given parts of the class definition. However, if necessary, you may define additional public and private data members and member functions in your class. You can also define additional classes in your solution. You are not allowed to use any global variables.

5. 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.

6. In this assignment, 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 "`PhoneBook`" and your files' name MUST BE "`PhoneBook.h`" and "`PhoneBook.cpp`". You should upload these two files (and any additional files if you wrote additional classes in your solution) as a single archive file (e.g., zip, tar, rar). The submissions that do not obey these rules will not be graded.

7. 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, do not submit a file that contains a function called `main`.

8. 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.

9. 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. 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.

10. This homework will be graded by your TA Nabil AbuBaker (nabil.abubaker at bilkent edu tr). Thus, you may ask your homework related questions directly to him.