CS 1410 Introduction to Computer Science – CS2
Section 1: MWF 10:30 a.m. – 11:20 a.m.
Section 2: MWF 1:00 p.m. – 1:50 p.m.
Instructor: Xiaojun Qi
Assignment #1

Given: Thursday, January 9, 2014
Due: 11:59 p.m. Saturday, January 18, 2014
Total Points: 30 points

Write a C++ program to manage a small next-generation social networking application. This C++ program keeps track of up to **five** online friends of yours. Initially, the "EMPTY" information is stored for each of the five friends. In other words, the Screen Name of each friend is EMPTY, the interests of each friend are EMPTY, and the age of each friend is 0.

Provide the user with the following menu interface:

***** Network of My Friends *****
A: Add a Friend
R: Remove a Friend
S: Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection:

You should write **three files (e.g., Friend.h, Friend.cpp, and Prog1.cpp) to solve the problem.**

1. Friend.h file should contain the following:
    - The definition of **a Friend structure**, which consists of three pieces of information: Screen Name, Interests, and Age. [Refer to chapter 8.12 (PDF version of the textbook) for materials on arrays of class objects (p. 549), arrays of structures (p. 555), etc.]
    - The function prototypes for the operations performed on Friend.
2. Friend.cpp file should contain the actual code (implementation) of each function listed in your Friend.h file.
3. Prog1.cpp file should be implemented using the available Friend manipulation functions in Friend.cpp and other appropriate functions.

The Friend.cpp file should at least contain the implementation of the following functions:

- **[3 points]** void AddFriend (Friend friends[], int size) ;
  Add a friend to the database at the first spot which contains the "EMPTY" information

- **[3 points]** void RemoveFriend (Friend friends[], int size) ;

Display the screen name of all the friends together with their locations (indices). Prompt the user to choose an appropriate index which corresponds to the friend to be deleted. Delete the chosen friend in the database by setting his/her screen name to "EMPTY", his/her interests to "EMPTY", and his/her age to 0.

- **[6 points]** void SearchInterest (Friend friends[], int size, string keywords) ;
  Search through the database to find friends whose interests match with the input keywords (**not case sensitive**) and output the friends' screen names.
  Hint:
  1) Convert the string of interests to contain all small letters or all capital letters using tolower or toupper functions (Refer to p. 797 of the PDF version of the textbook for the usage of these two functions provided by C++ library). Make sure to include cctype.
  2) Check out Table 12-8 (p. 809 of the PDF version of the textbook) to find the appropriate member function to do the matching task. Or use the loop to do the matching task by yourself.

- **[3 points]** void DisplayFriend (Friend friends[], int size) ;
  Sequentially list the information of all the friends in the database together with their indices.

- **[6 points]** void ListFriend(Friend friends[], int size) ;
  Alphatically list the screen name of all the friends in the database.
  Note/Hint:
  You should not change the content of your friend database after calling this function. To accomplish this task, you had better declare another temporary array with the size of 5. This temporary array will be initialized first. You can then sequenitally copy your friends' information from your database by skipping the "EMPTY" friend. Finally, you perform the sorting operation on this temporary array by passing down the number of your friends, which may be less than 5. In this way, the "EMPTY" friend will not be sorted.

- **[3 points]** float ReportAge(Friend friends[], int size) ;
  Report the average age of all friends.

- bool IsBefore (Friend friend1, Friend friend2) ;
  Implement the **IsBefore(Friend A, Friend B)** function with the following criteria:
  1. Friend A is before Friend B if A's screen name is alphabetically before B's.
  2. If their screen names are the same, compare their ages.
  3. If screen names and ages are the same, it doesn't matter who comes first.

Below is the sample code for Bubble sort for Friend, which is similar to the Bubble sort you learned in class except that the Friend ADT is used.

```
void BubbleSort (Friend array[ ], int size)
{
    bool done = false;
    while (!done)
    {
        done = true;
        for (int n=0; n<size-1; n++)
          if (! IsBefore (array[n], array[n+1]) )
          {
              Swap (array[n], array[n+1]);
              done = false;
          }
    }
    size -- ;
}

void Swap (Friend &a, Friend &b)
{
        Friend temp ;
        temp = a ;
        a = b ;
        b = temp ;
}
```

Note: BubbleSort function should be called inside ListFriend function to sort the screen names in the alphatical order. You can copy the above two functions in your program. However, you have to implement IsBefore function to accomplish the sorting task.

For your quick reference, I list sample runs of the program below:

```
***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: A
*** Add a new friend profile
Screen Name: QiTheGreat
Interests: Movie, Research, Travel
Age: 35

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit
```

```
Selection: A
*** Add a new friend profile
Screen Name: QiTheBoss
Interests: Programming, Teach, TRAVEL
Age: 50

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: D

Index 0:
Screen Name: QiTheGreat
Interests: Movie, Research, Travel
Age: 35

Index 1:
Screen Name: QiTheBoss
Interests: Programming, Teach, TRAVEL
Age: 50

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: S
Search Keyword: research
*** QiTheGreat

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: s
Search Keyword: travel
*** QiTheGreat
*** QiTheBoss

***  Network of My Friends ***
A: Add a Friend
```

```
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: L
QiTheBoss
QiTheGreat

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: R
***      Remove a friend ***
0. QiTheGreat
1. QiTheBoss

Which to Remove: 0

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: D
Index 1:
Screen Name: QiTheBoss
Interests: Programming, Teach, TRAVEL
Age: 50

***  Network of My Friends ***
A: Add a Friend
R: Remove a Friend
S:  Search Interests
D: Display All Friends
L: List All Friends in Alphabetic Order Based on ScreenName
C: Calculate the Average Age of All Friends in My Network
E: Exit

Selection: c
The average age of all friends in my network is 50.
```