CS 1410 Introduction to Computer Science – CS2
Section 1: MWF 9:30 a.m. – 10:20 a.m.
Instructor: Xiaojun Qi
Assignment #8

Given: Friday, March 8, 2013
Due: 11:59 p.m. Saturday, March 23, 2013
Total Points: 30 points

Databases provide useful information. Usually the records in a database are accessed by means of an identification number. The simplest databases consist of information stored in a file. The information is read into memory from the file at the time that the access software starts.

Your C++ program will read information, representing an online friend of your social network (see Assignment #2), from a binary file named myNetwork.dat, that is located in the same directory as the C++ program. This file has been created in the format of a .NET Visual C++ binary file. The file contains some (indefinite) number of blocks of data (structures), each representing a friend of your social network in the following format:

- Screen name – a C-string containing 30 characters
- Interests – a C-string containing 100 characters
- Age – a short integer

The program functions as follows:

1. Print the menu that lists the user commands:

   ***** Network of My Friends *****
   A: Add a Friend
   D: Delete a Friend by ID Number
   M: Modify a Friend's Information by ID Number
   S: Search Interests
   PD: Print "myNetwork.dat" Database Information
   PO: Print the Oldest and the Youngest Friends' Information
   E: Exit

2. Perform the command specified.
3. Reprint the menu.
4. Repeat steps 2 through 4 until the user chooses to exit.

Here, *each friend has a unique ID number. This ID number is also the file record or object number of the friend.* The file record number for your friends starts with one and continues consecutively to the end. Thus, if you have ten friends, the record numbers will be from one to ten. The record number is not given by the user, but given by the program. The record numbers are not part of the Friend structure.

Your C++ program keeps track of your online friends stored in a binary file, "myNetwork.dat". Initially, your program prepares (creates) a binary file "myNetwork.dat", that is located in the same directory as the program, for possible future reading and writing operation. Specifically, the program opens a binary file "myNetwork.dat" for the reading operation. If "myNetwork.dat" does not exist, the program will create a new blank "myNetwork.dat" file. Otherwise, the program will prompt the user the following information:

```
"myNetwork.dat" already exists.  Overwrite file?  (Y/N)
```

If the user types 'Y' or 'y', the program will erase all the contents in "myNetwork.dat". If the user types 'N' or 'n', the program will keep the original content intact. This "myNetwork.dat" file can hold ***any number*** of Friend objects or records.

Your main program should be implemented using the available Friend manipulation functions in Friend.cpp. That is, all functions related to Friend should be implemented in Friend.cpp and main program should call these functions or other functions to accomplish the task. Please refer to Chapter13Example.zip to understand where to put each function and how every function is called in the main program. ***At one given time, your program should only hold a single Friend record or object* – rather than a large array.** The **only** arrays that may be used are for C-strings. **You may not use vectors or arrays in this assignment!**

The options listed in the menu should have the following effects:

A: **This option should not ask the user for an ID number.** Instead, it should report to the user the unique ID that the *program* has just assigned. This unique ID should be the *file record or object number*. Add the new Friend's information to the right position of the "myNetwork.dat" file.

D: Delete the Friend object specified by the unique ID from the "myNetwork.dat" file. Please provide the validation process to ensure that the input ID number is a valid one. That is, give an appropriate error message if an invalide ID is given (out of the range). In the meantime, provide the following confirmation message before deleting the Friend object from the disk file.

> The indicated Friend object information is:
> xxxx (Here: xxx should correspond to the actual Friend object information)
> Do you intend to delete the indicated Friend object from the disk file? Press Y to delete this Friend object. Press N to cancel this deletion.
> Note: **Once the specified friend is deleted, all friend records after this deleted friend should be moved forward by one record.** For example, if the deleted friend is No. 3, the No. 4 friend automatically becomes No. 3 friend, the No. 5 friend automatically becomes No. 4 friend, etc.

M: Change the indicated Friend object information in the "myNetwork.dat" file with different information. Please provide the validation process to ensure that the input ID number is a valid one. That is, give an appropriate error message if an invalide ID is given (out of the range). In the meantime, provide the following confirmation message:

> The indicated friend object information is:

xxxx (Here: xxx should correspond to the actual Friend object information)
The new updated friend object information is:
yyyy (Here: yyy should correspond to the new information, which is gathered from the user's input)
Do you intend to replace the indicated friend object with this new information? Press Y to replace this information in the database. Press N to cancel this update.

S: Search through the database to find friends whose interests match with the input keyword (case insensitive) and output the friends' screen name together with his/her ID number.

PD: Provide a summary report of all actual Friend objects in "myNetwork.dat".

PO: Provide a summary report of the oldest and youngest Friend objects in "myNetwork.dat" in order. That is, the oldest friends are listed before the youngest friends.

E: Close the opened file "myNetwork.dat".

A Friendly Reminder:
When you've gone all the way through a file, make sure you execute the following command binary.clear( );, where binary is the variable name for the binary file. This statement will clear the error state flag eofbit, which is set when the end of file is encountered. So the seekp() and seekg() operations can be performed again.