EE 355 Project Phase 1 Readme

Name: Tina Niknejad
USC ID: 6361622359
Email: tnikneja@usc.edu

Name: Jeff Cui
USC ID: 8181687359
Email: jeffcui@usc.edu

Program Summary:
For this program we implemented four classes: Date, Contact, Person, and Network, to create a social network that uses a doubly linked list to store information about the people inside of it. Each person has information on their first name, last name, birthdate, email type, email, phone type, and phone number. Birthdate is stored as class Date, email and phone are two children classes of class Contact. Class Person stores all of the info and class Network has a doubly linked list that holds/connects all of the people and creates the user interface.

The class Date provides us with functions that check date, set date, get date, and print date. There is also a constructor that sets the date every time the class is called. The class Contact has functions that set and get information for type and corresponding contents of email and phone number. It also has email and phone constructors that get the type and corresponding data of the email and phone number in the form of a string.

Class Person sets five attributes: first name, last name, birthday, email type and email, and phone type and phone number. It sets attributes in three ways- one takes a file as an input and the other two take user inputted data (difference is one is called from Person and the other is usually called from Network when adding a Person to it). Person also has two constructors. One has direct inputs for all attributes and the other takes a file as the input. The destructor deletes all dynamically allocated memory associated with the Person class such as email, phone number, and birthday. Lastly, the person class has boolean operators that check whether or not one person is equal to another based on first name, last name, and birthday.

Class Network brings everything together. Class Network prints the database using the function printDB, loads the network using the function loadDB, saves the network to a file using the function saveDB, adds a person if they do not already exist in the database using the function push_front, searches for a person using the function search, and removes a person using the function remove. Network also has various other functions like constructors that handle functionality.

References:
Used slides and examples from 355, mainly ones on OOP, linked lists and STL

For date.cpp
This pointer:
https://www.geeksforgeeks.org/this-pointer-in-c/

For contact.cpp
Protected/Private/Public inheritance:
https://stackoverflow.com/questions/860339/difference-between-private-public-and-protected-inheritance

Substring:
http://www.cplusplus.com/reference/string/string/substr/

Default arguments (to handle if style == “full”):
https://www.geeksforgeeks.org/default-arguments-c/

For person.cpp
Get length of a string:
https://stackoverflow.com/questions/905355/how-to-get-the-number-of-characters-in-a-stdstring

Insert character into string using push_back:
http://www.cplusplus.com/reference/string/string/push_back/

Operator Overloading:
https://www.geeksforgeeks.org/operator-overloading-c/

For network.cpp
Opening Files with c_str():
https://stackoverflow.com/questions/6323619/c-ifstream-error-using-string-as-opening-file-path
http://www.cplusplus.com/reference/string/string/c_str/

Dirent/Finding .db files:
https://pubs.opengroup.org/onlinepubs/007908799/xsh/dirent.h.html
https://stackoverflow.com/questions/24447540/c-how-to-search-files-in-a-directory-with-certain-name
Vectors:
http://www.cplusplus.com/reference/vector/vector/

Adding a node to a Doubly Linked List:
https://www.geeksforgeeks.org/doubly-linked-list/

To overwrite a text file when writing to it:
http://www.cplusplus.com/forum/beginner/195138/

To delete a node in a Doubly Linked List:
https://www.geeksforgeeks.org/delete-a-node-in-a-doubly-linked-list/

General googling/references from GeeksforGeeks and cplusplus reference were used to handle small questions
----

Instructions:

1. Make sure all necessary files are in the same directory: all .cpp and .h files, the test_network.cpp file, .db files and an empty txt file created to save databases.
2. Run/compile all .cpp files together into an executable.
3. Run the executable and there are 6 options each with functionality and uses.

Other Information:
A string was used to save phone numbers instead of a number variable like int, long, long long, etc. This is because it is easier to handle with user inputs and most of the time the phone number has to be handled as a string anyway, so converting between variable types is not very efficient.

There are some check statements here and there to make sure things like user input don't break the code, etc. However, for the most part the code is built to assume that the user will input the correct format and info when prompted every time. Most attributes are handled to allow any input except the phone number, which has to be inputted precisely. It can be handled with dashes or without.

The creation of static variables may not be the most memory efficient, but was ensured that the program worked easily/intuitively and to also make the code easy to follow. However, dynamic memory and memory leak was checked and accounted for when coding the project. To do comments were also not removed to make it easier to see the code that was added or modified. All parts/functions/separate .cpp files were checked individually before inserting it into the larger system. Some of these tests and cout statements to check may be commented out in the code.