Phase 1: TrojanBook Core Functionality

This document provides an explanation of the implementation of Phase 1 of the EE 355 Project: A contact management system that resembles a simple social network (TrojanBook).

Implemented Classes

1. **`Contact` Class (`contact.h`, `contact.cpp`)**
   - Abstract base class with two derived classes: `Email` and `Phone`.
   - Implementation includes virtual methods for setting contact info, getting contact info, and printing.
   - `type` attribute made `protected` to allow derived classes to access it.
   - `Email` class stores and manages email address information.
   - `Phone` class handles phone numbers with proper formatting (`XXX-XXX-XXXX`).
2. **`Date` Class (`date.h`, `date.cpp`)**
   - Handles date information with different format options.
   - Implemented parsing of `M/D/YYYY` format.
   - Added `print_date` method with formatting options: `MM/DD/YYYY` and `Month D, YYYY`.
   - Added comparison operators (`==`, `!=`) for date equality checking.
3. **`Person` Class (`person.h`, `person.cpp`)**
   - Stores personal information: first name, last name, birthdate, email, and phone.
   - Implements constructors, including one that reads from file.
   - Added support for doubly linked list pointers (`next`, `prev`).
   - Implemented comparison operators for person equality.
   - Proper memory management in destructor for dynamic objects (`Date`, `Email`, `Phone`).
4. **`Network` Class (`network.h`, `network.cpp`)**
   - Manages a doubly linked list of `Person` objects.
   - Implemented load (`loadDB`) and save (`saveDB`) functionality for database files.
   - Added `search` methods to find people by name or by `Person` object.
   - `push_front` and `push_back` methods for adding people to the network.
   - `remove` method to delete people from the network.
   - `showMenu` method to provide an interactive text-based user interface.

Key Design Decisions

- **Memory Management:** All dynamically allocated objects (`Date`, `Email`, `Phone`) are properly deleted in `Person`'s destructor. The `Network` destructor properly cleans up all `Person` objects in the linked list.

- **Data Storage:** Phone numbers stored as strings with formatting applied. Used `protected` inheritance for the `type` attribute in `Contact` class.
- **File I/O:** Implemented robust file reading/writing with error handling. Supports both single person files and network database files (`networkDB.txt` format).
- **User Interface:** Interactive menu (`showMenu`) with clear prompts and basic error handling.

Testing

The Phase 1 implementation was tested with:

- Creating and manipulating individual `Person` objects.
- Loading and saving from/to files (`loadDB`, `saveDB`).
- Adding (`push_front`, `push_back`) and removing (`remove`) people from the network.
- Searching (`search`) for people by name.
- Testing equality between `Person` objects (`operator==`).

Compilation Instructions

```
# To compile the Person equality test:
g++ -o test_person date.cpp contact.cpp person.cpp test_person_eq.cpp

# To compile the Network interactive test:
g++ -o test_network date.cpp contact.cpp person.cpp network.cpp misc.cpp test_network.cpp
```

Future Enhancements (Considered after Phase 1)

- Better input validation.
- More robust error handling.
- Supporting multiple contact information per person.
- Implementing additional search and sort options.
- Graphical user interface.