

Lab 3 - Manage Contacts

4/6/2024

0/50 Points

Attempt 1



Review Feedback

Offline Score:

0/50



Add Comment

Anonymous Grading: no

Unlimited Attempts Allowed

3/11/2024

Details

Create an object-oriented programming (OOP) program that manages student contact.

Like your lab 2, in this lab exercise, you will learn to organize, store, and perform operations on data using linked lists.

**** You may use the existing implementation in Lab 2, *utils*, *menu*, *datetime*, etc.**

Lab 3 Requirements:

1) Implement a class `Person` that contains the following data members:

```
string first_name
string middle_name
string last_name
DateTime birthdate
string address
string city
string county    // Ex. Alameda
string state
string zip
string phone1    // Primary
string phone2    // Secondary
string pronouns  // he/she
string email
```

2) Implement a new class `Contact`: `public Person` - be sure to implement all the necessary data, methods, and operators

See the data file [contact_data.csv \(https://ohlone.instructure.com/courses/29469/files/5030421?wrap=1\)](https://ohlone.instructure.com/courses/29469/files/5030421?wrap=1)
 [\(https://ohlone.instructure.com/courses/29469/files/5030421/download?download_frd=1\)](https://ohlone.instructure.com/courses/29469/files/5030421/download?download_frd=1) for the data structures of a contact.

Example:

```
id          // contact id
role        // instructor, student, etc.
company     // company name or organization name
```


3) Implement a new base class template `LinkedList` and the necessary data, methods, and operators, and sort the list of contacts

- Implement the necessary data, methods, operators

See [selectionSort-example-3-12-2024.zip \(https://ohlone.instructure.com/courses/29469/files/5202696?wrap=1\)](https://ohlone.instructure.com/courses/29469/files/5202696?wrap=1)
 [\(https://ohlone.instructure.com/courses/29469/files/5202696/download?download_frd=1\)](https://ohlone.instructure.com/courses/29469/files/5202696/download?download_frd=1)

4) Add a new derive class `ContactList`: `public LinkedList`

5) Implement a member method `ContactList::sortBy <any fieldname>` using `selectionSort`

See [linkedList-example-3-12-2024.zip \(https://ohlone.instructure.com/courses/29469/files/5202697?wrap=1\)](https://ohlone.instructure.com/courses/29469/files/5202697?wrap=1). 
https://ohlone.instructure.com/courses/29469/files/5202697/download?download_frd=1

6) Implement a member method `ContactList::search` contact in the list

7) Implement a new class `ContactMenu`: public `Menu` like your class `UserMenu` from Lab 1

Example:

***** Contact Menu *****

- 1) List of contacts
- 2) View a contact
- 3) Add new contact
- 4) Edit contact
- 5) Delete contact
- x) Exit

8) Implement a menu select to navigate the record move first, move previous, move next, move last

From the user menu, the user selects "List of Contacts", your program should be able to selectionSort any field (ascending, descending), and perform record navigation.

You will need to add, the following member methods

```
class ContactList {
    void moveFirst();
    void moveNext();
    void movePrevious();
    void moveLast();
}
```

9) Implement the pre-increment operator and decrement operator. Your operator increments (++) for moving to the next record and operator decrement (--) for the previous for moving to the previous record.

```
// prefix increment
Iterator<T>& operator++()
// postfix increment
Iterator<T> operator++(T)
// prefix decrement
Iterator<T>& operator--()
// postfix decrement
Iterator<T> operator--(T)
```

9) Implement the following operations for your `ContactList`

From the contact menu, implement these methods and handlers

User select:

- 1) List of Contacts
 - `doList()` - Show the sorted list of contacts in either ascending or descending order
 - `doSortBy()` - Implement the selection sorting by <any fieldname> in the contact and list the contacts
- 2) View a contact
 - `doView()` - Select a contact from the list and show the contact details
- 3) Add new contact
 - `doAdd()` - Add new contact
- 4) Edit contact
 - `doEdit()` - Select a contact to edit an existing contact

5) Delete contact

- doDelete() - Select a contact to eliminate existing contact

x) Exit

- doExit() - Save data to disk (contact_data.csv) and exit the contact menu

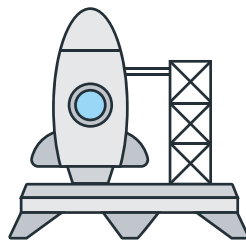
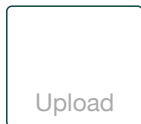
10) Use <iomanip> to format the output for cout and follow the C++ Programming guidelines

Submission: lab3-<your name>.zip and the screenshots of the output.

Importantly: Your lab assignment is to be done individually. You may discuss the concepts with other students in class. You may not copy someone else's work.

[\(\\$CANVAS COURSE REFERENCES/file_ref/g956448d4a96f07fc7bcc905b88e01618?wrap=1\)](#)

Choose a submission type



Choose a file to upload

or

Webcam Photo

Canvas Files



<https://ohlone.instructure.com/courses/29469/modules/items/1488936>



<https://ohlone.instructure.com/courses/29469/modules/items/>