Computing 2 – Labs

Lab 5: Class Directory and Class Entry

This example covers two classes and uses dynamic memory allocation. The classes are `Entry` and `Directory`. Most of the code has already been provided (see EntryClass.zip and DirectoryClass.zip).

`Entry`:

An object of the class `Entry` represents a single entry in a phone book. The data members stored in an entry object are name, address, and phone number. Strings are used to store these. You are provided the files entry.h and entry.cpp. Please do the following:

1. Finish the implementation of the member function Entry::`requestEntryFromUser()`. To this end, use the function `getline(…)` as this takes a complete line of user entries from `cin`, even if separate entries are separated by white space.
2. Complete the driver program to test the class (see main.cpp).
3. Change the default constructor in entry.h (and entry.cpp) such that the arguments name, phone_number, and address are now passed by `const` reference (and not by value as currently the case).

The user interaction of your driver program may look like this:

```
Type name, followed by pressing RETURN key: Smith
Type phone number, followed by pressing RETURN key: 001/650/498-1234
Type address, followed by pressing RETURN key: University Ave., Palo Alto
Name: Smith, Phone Number: 001/650/498-1234, Address: University Ave., Palo Alto
```

`Directory`:

An object of type `Directory` stores a list of `Entry` objects, using a dynamically allocated array. The `Directory` class also provides services (public member functions) for adding new entries and displaying all entries in the phone book. The `Directory` class also has a private helper function, `doubleMaxSize()`, for dynamically resizing the array of Entries when more memory space is needed.

There is a member variable `max_size` specifying the initial maximum size for the number of entries. Its default value is five. When adding an entry to the directory its member variable `current_size` is to be incremented. If the `current_size` reaches `max_size`, then more memory needs to be allocated. In our example, twice the memory is allocated.

Note that the destructor is also implemented for the `Directory` class. Since an array was allocated, the destructor needs to be called as (note the [])

```
delete [] entry_list_ptr_;
```

This deallocates the dynamic array pointed to by `entry_list_ptr`.

Tasks:

1. Add include guards to directory.h
2. Modify the destructor according to the instructions above.
3. Complete implementation of the member function `Directory::insertEntry()`
4. Finish the implementation of the member function `Directory::displayDirectory()`

If your implementation was successful, your program may work like this:

```
                *** DIRECTORY ***
        i       Insert a new entry into the directory
        d       Display the entire directory
        q       Quit
i

Type name, followed by pressing RETURN key: Test1

Type phone number, followed by pressing RETURN key: 001

Type address, followed by pressing RETURN key: Address1

                *** DIRECTORY ***
        i       Insert a new entry into the directory
        d       Display the entire directory
        q       Quit
i

Type name, followed by pressing RETURN key: Test2

Type phone number, followed by pressing RETURN key: 002

Type address, followed by pressing RETURN key: Address2

                *** DIRECTORY ***
        i       Insert a new entry into the directory
        d       Display the entire directory
        q       Quit
i

Type name, followed by pressing RETURN key: Test3

Type phone number, followed by pressing RETURN key: 003

Type address, followed by pressing RETURN key: Address3

                *** DIRECTORY ***
        i       Insert a new entry into the directory
        d       Display the entire directory
        q       Quit
d
Directory Entries
Name: Test1      Phone Number: 001       Address: Address1
Name: Test2      Phone Number: 002       Address: Address2
Name: Test3      Phone Number: 003       Address: Address3

                *** DIRECTORY ***
        i       Insert a new entry into the directory
        d       Display the entire directory
        q       Quit
q

Freeing up memory pointed to by entry_list_ptr_
```