```
Script started on 2023-12-07 17:55:44-06:00 [TERM="xterm" TTY="/dev/pts/15" COLUMNS
a_vitale3@ares:~$ pwd
/home/students/a_vitale3
a_vitale3@ares:~$ cat NamesAndAddr.
Name: Andrew Vitale

Class: CSC121



Activity: Names And Addresses

Level: 6



Description:



Allows a user to input a first and last name, address, email, and phone number.

this program also allows the user to edit, delete, search, and list all info.

a_vitale3@ares:~$ show-code NamesAndAddr.cpp


NamesAndAddr.cpp:
```

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <limits>

#include "rolodex.h"

using namespace std;

void interface();
void sub_interface(string text);
void listEntries(bool list);

vector<RolodexEntry> entries;

int main()
{
    bool exit = false;
    int input_int = 0;
    long unsigned int index;
    string input_str;
    do
    {
        interface();
        cin >> input_str;
        if (input_str.find('1') != string::npos)
        {
            RolodexEntry entry;
            entry.input();
            entries.push_back(entry);
        }
        else if (input_str.find('2') != string::npos)
        {
            while(true)
            {
                sub_interface("edit");
                cin >> input_str;
                if (input_str.find('1') != string::npos)
                {
                    listEntries(true);
                    cout << "Edit the name of entry number: ";
                    std::cin.ignore(std::numeric_limits<std::streamsize>::r
                    if (isdigit(std::cin.peek()))
                    {
                        cin >> input_int;
                    }
                    else
                    {
                        std::cout << "Error.\n";
                        input_int = 0;
                    }
                    input_int -= 1;
                    if (input_int >= 0 && static_cast<long unsigned int>(i
                    {
                        cout << "New first name of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
                        cout << "New last name of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
                    }
                }
                else if (input_str.find('2') != string::npos)
                {
                    listEntries(true);
                    cout << "Edit the address of entry number: ";
                    std::cin.ignore(std::numeric_limits<std::streamsize>::r
                    if (isdigit(std::cin.peek()))
                    {
                        cin >> input_int;
                    }
                    else
                    {
                        std::cout << "Error.\n";
                        input_int = 0;
                    }
                    input_int -= 1;
                    if (input_int >= 0 && static_cast<long unsigned int>(i
```

```
                    {
                        cout << "New street of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
                        cout << "New last town of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
                        cout << "New last state of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
                    }
                }
                else if (input_str.find('3') != string::npos)
                {
                    listEntries(true);
                    cout << "Edit the phone number of entry number: ";
                    std::cin.ignore(std::numeric_limits<std::streamsize>::r
                    if (isdigit(std::cin.peek()))
                    {
                        cin >> input_int;
                    }
                    else
                    {
                        std::cout << "Error.\n";
                        input_int = 0;
                    }
                    input_int -= 1;
                    if (input_int >= 0 && static_cast<long unsigned int>(i
                    {
                        cout << "New phone number of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
                    }
                }
                else if (input_str.find('4') != string::npos)
                {
                    listEntries(true);
                    cout << "Edit the email of entry number: ";
                    std::cin.ignore(std::numeric_limits<std::streamsize>::r
                    if (isdigit(std::cin.peek()))
                    {
                        cin >> input_int;
                    }
                    else
                    {
                        std::cout << "Error.\n";
                        input_int = 0;
                    }
                    input_int -= 1;
                    if (input_int >= 0 && static_cast<long unsigned int>(i
                    {
                        cout << "New email of entry: ";
                        cin >> input_str;
                        entries[static_cast<long unsigned int>(input_int)]
```

```
                    }
                }
                else
                {
                    break;
                }
            }
        }
        else if (input_str.find('3') != string::npos)
        {
            listEntries(true);
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\
            if (isdigit(std::cin.peek()))
            {
                cin >> input_int;
            }
            else
            {
                std::cout << "Error.\n";
                input_int = 0;
            }
            input_int -= 1;
            if(input_int >= 0 && static_cast<long unsigned int>(input_int)
            {
                entries.erase(entries.begin()+static_cast<long unsigned int
            }
        }
        else if (input_str.find('4') != string::npos)
        {
            while (true)
            {
                sub_interface("search by");
                cin >> input_str;
                if (input_str.find('1') != string::npos)
                {
                    cout << "Entry name: ";
                    cin >> input_str;
                    index = 0;
                    for (auto entry : entries)
                    {
                        if (entry.get_firstName().find(input_str) != string
                        {
                            cout << "Entry is at postion " << index + 1 <<
                        }
                        else
                        {
                            index += 1;
                        }
                    }
                    if (index >= entries.size())
                    {
                        cout << "Entry does not exist." << endl;
                    }
                }
```

```
            else if (input_str.find('2') != string::npos)
            {
                cout << "Entry address: ";
                cin >> input_str;
                index = 0;
                for (auto entry : entries)
                {
                    if (entry.get_street().find(input_str) != string::n
                    {
                        cout << "Entry is at postion " << index + 1 <<
                    }
                    else
                    {
                        index += 1;
                    }
                }
                if (index >= entries.size())
                {
                    cout << "Entry does not exist." << endl;
                }
            }
            else if (input_str.find('3') != string::npos)
            {
                cout << "Entry phone number: ";
                cin >> input_str;
                index = 0;
                for (auto entry : entries)
                {
                    if (entry.get_phone().find(input_str) != string::np
                    {
                        cout << "Entry is at postion " << index + 1 <<
                    }
                    else
                    {
                        index += 1;
                    }
                }
                if (index >= entries.size())
                {
                    cout << "Entry does not exist." << endl;
                }
            }
            else if (input_str.find('4') != string::npos)
            {
                cout << "Entry email: ";
                cin >> input_str;
                index = 0;
                for (auto entry : entries)
                {
                    if (entry.get_email().find(input_str) != string::np
                    {
                        cout << "Entry is at postion " << index + 1 <<
                    }
                    else
                    {
                        index += 1;
                    }
                }
                if (index >= entries.size())
                {
                    cout << "Entry does not exist." << endl;
                }
            }
            else if (input_str.find('5') != string::npos)
            {
                if (entries.size() != 0)
                {
                    listEntries(false);
                }
                else
                {
                    cout << "You have not input any entries.\n";
                }
            }
            else if (input_str.find('6') != string::npos)
            {
                exit = true;
            }
        } while (exit != true);
        return 0;
    }

    void interface()
    {
        cout << "1) Add entry\n" <<
            "2) Edit entry\n" <<
            "3) Delete entry\n" <<
            "4) Search for entry\n" <<
            "5) Print all entries\n" <<
            "6) Quit\n";
    }

    void sub_interface(string text)
    {
        cout << "1) " << text << " Name\n" <<
            "2) " << text << " Address\n" <<
            "3) " << text << " Phone number\n" <<
            "4) " << text << " Email address\n" <<
            "5) return to Main Menu\n";
    }

    void listEntries(bool list)
```

```
{
    int incrementor = 0;
    for (auto entry : entries)
    {
        if (list == true)
        {
            cout << incrementor+1 << ": ";
            incrementor += 1;
            cout << entry.get_firstName() << " " << entry.get_lastName() <<
        }
        else
        {
            entry.output();
        }
    }
    cout << endl;
}
```

```
a_vitale3@ares:~$ show-code NamesAndAddolodex
```

rolodex.cpp:

```
#include "rolodex.h"

#include <iostream>
#include <string>
#include <limits>

// constructor's, accessors, mutators, i/o, equality, etc.

//constructor
RolodexEntry::RolodexEntry()
{
    set_firstName("first name");
    set_lastName("last name");
    set_street("street");
    set_town("town");
    set_state("state");
    set_zip(0);
    set_zip_4(0);
    set_phone("phone");
    set_email("email");
    return;
}
RolodexEntry::RolodexEntry(std::string text_one, std::string text_two)
{
    set_firstName(text_one);
    set_lastName(text_two);
    set_street("street");
    set_town("town");
    set_state("state");
    set_zip(0);
    set_zip_4(0);
    set_phone("phone");
    set_email("email");
    return;
}
/*RolodexEntry::RolodexEntry(std::string fname, std::string lname, std::str
{
    set_firstName(fname);
    set_lastName(lname);
    set_street(street);
    set_town(town);
    set_state(state);
    set_zip(zip);
    set_zip_4(zip_4);
    set_phone(phone);
    set_email(email);
    return;
}*/

//i/o
void RolodexEntry::input(void)
{
    std::cout << "\nFirst Name: ";
    std::cin >> fname;
    std::cout << "Last Name: ";
    std::cin >> lname;
    std::cout << "Street Name: ";
    std::cin >> street;
    std::cout << "Town Name: ";
    std::cin >> town;
    std::cout << "State Name: ";
    std::cin >> state;
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    std::cout << "Zip code: ";
    if (isdigit(std::cin.peek()))
    {
        std::cin >> zip;
    }
    else
    {
        std::cout << "Error.\n";
        zip = 0;
    }
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    std::cout << "Zip 4 code: ";
    if (isdigit(std::cin.peek()))
    {
        std::cin >> zip_4;
    }
    else
    {
        std::cout << "Error.\n";
        zip_4 = 0;
    }
    std::cout << "Phone Number: ";
```

```
    std::cin >> phone;
    std::cout << "Email: ";
    std::cin >> email;
}
void RolodexEntry::output(void) const
{
    std::cout << "\nFirst Name: "
        << fname << ", "
        << "Last Name: "
        << lname << ", "
        << "Street Name: "
        << street << ", "
        << "Town Name: "
        << town << ", "
        << "State Name: "
        << state << ", "
        << "Zip code: "
        << zip << ", "
        << "Zip 4 Code: "
        << zip_4 << ", "
        << "Phone Number: "
        << phone << ", "
        << "Email: "
        << email << ".";
}
void RolodexEntry::edit(void)
{
    std::string input;
    std::cout << "Would you like to edit First Name? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "\nFirst Name: ";
        std::cin >> fname;
    }
    std::cout << "Would you like to edit Last Name? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Last Name: ";
        std::cin >> lname;
    }
    std::cout << "Would you like to edit Street Name? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Street Name: ";
        std::cin >> street;
    }
    std::cout << "Would you like to edit Town Name? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Town Name: ";
        std::cin >> town;
    }
    std::cout << "Would you like to edit State Name?";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "State Name: ";
        std::cin >> state;
    }
    std::cout << "Would you like to edit Zip code? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Zip code: ";
        std::cin >> zip;
    }
    std::cout << "Would you like to edit Zip 4 code? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Zip 4 code: ";
        std::cin >> zip_4;
    }
    std::cout << "Would you like to edit Phone Number? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Phone Number: ";
        std::cin >> phone;
    }
    std::cout << "Would you like to edit Email? ";
    std::cin >> input;
    if (input == "y" || input == "yes")
    {
        std::cout << "Email: ";
        std::cin >> email;
    }
}

//equality
bool RolodexEntry::operator==(const RolodexEntry rol2)
{
    if (fname == rol2.fname &&
        lname == rol2.lname &&
        street == rol2.street &&
        town == rol2.town &&
        state == rol2.state &&
        zip == rol2.zip &&
        zip_4 == rol2.zip_4 &&
        phone == rol2.phone &&
        email == rol2.email)
    {
        return true;
    }
```

```
    else
    {
        return false;
    }
}
a_vitale3@ares:~$ show-code rolodex.h


rolodex.h:


#ifndef ROLODEX_HELPER_LIBRARY_INCLUDED
#define ROLODEX_HELPER_LIBRARY_INCLUDED

#include <string>

class RolodexEntry
{
    std::string fname = "first name", lname = "last name";  // together or
    std::string street = "street", town = "town", state = "state";
    long zip = 0;
    short zip_4 = 0;
    std::string phone = "phone",  // or 3 short's?  (area, exchange, line)
        email = "email";
public:
    // constructor's, accessors, mutators, i/o, equality, etc.

    //constructor
    RolodexEntry(void);
    RolodexEntry(std::string text_one, std::string text_two);
    //RolodexEntry(std::string fname, std::string lname, std::string street

    //i/o
    void input(void);
    void output(void) const;
    void edit(void);

    //accessors
    std::string get_firstName() const { return fname; }
    std::string get_lastName() const { return lname; }
    std::string get_street() const { return street; }
    std::string get_town() const { return town; }
    std::string get_state() const { return state; }
    long get_zip() const { return zip; }
    short get_zip_4() { return zip_4; }
    std::string get_phone() const { return phone; }
    std::string get_email() const { return email; }

    //mutators
    void set_firstName(const std::string text) { fname = text; return; }
    void set_lastName(const std::string text) { lname = text; return; }
    void set_street(const std::string text) { street = text; return; }
    void set_town(const std::string text) { town = text; return; }
    void set_state(const std::string text) { state = text; return; }
    void set_zip(const long num) { zip = num; return; }
    void set_zip_4(const short num) { zip_4 = num; return; }
    void set_phone(const std::string text) { phone = text; return; }
    void set_email(const std::string text) { email = text; return; }

    //equality
    bool operator==(const RolodexEntry rol2);

};

#endif
a_vitale3@ares:~$ CPP NamesAndAddr rolodex rolodex.h
NamesAndAddr.cpp***
rolodex.cpp...


a_vitale3@ares:~$ ./NamesAndAddr.out
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
1

First Name: first
Last Name: last
Street Name: street
Town Name: town
State Name: state
Zip code: 1111
Zip 4 code: 1111
Phone Number: 111-111-1111
Email: email@gmail.com
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
5

First Name: first, Last Name: last, Street Name: street, Town Name: town, State Nar
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
2
1) edit Name
2) edit Address
3) edit Phone number
4) edit Email address
```

```
5) return to Main Menu
1
1: first last

Edit the name of entry number: 1
New first name of entry: newname
New last name of entry: newlast
1) edit Name
2) edit Address
3) edit Phone number
4) edit Email address
5) return to Main Menu
2
1: newname newlast

Edit the address of entry number: 1
New street of entry: newstreet
New last town of entry: newtown
New last state of entry: newstate
1) edit Name
2) edit Address
3) edit Phone number
4) edit Email address
5) return to Main Menu
3
1: newname newlast

Edit the phone number of entry number: 1
New phone number of entry: 333-333-3333
1) edit Name
2) edit Address
3) edit Phone number
4) edit Email address
5) return to Main Menu
4
1: newname newlast

Edit the email of entry number: 1
New email of entry: newenail@hotmail.com
1) edit Name
2) edit Address
3) edit Phone number
4) edit Email address
5) return to Main Menu
5
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
5

First Name: newname, Last Name: newlast, Street Name: newstreet, Town Name: newtow
```

```
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
4
1) search by Name
2) search by Address
3) search by Phone number
4) search by Email address
5) return to Main Menu
1
Entry name: newname
Entry is at postion 1
1) search by Name
2) search by Address
3) search by Phone number
4) search by Email address
5) return to Main Menu
2
Entry address: new
Entry is at postion 1
1) search by Name
2) search by Address
3) search by Phone number
4) search by Email address
5) return to Main Menu
3
Entry phone number: 3
Entry is at postion 1
1) search by Name
2) search by Address
3) search by Phone number
4) search by Email address
5) return to Main Menu
4
Entry email: new
Entry is at postion 1
1) search by Name
2) search by Address
3) search by Phone number
4) search by Email address
5) return to Main Menu
5
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
3
1: newname newlast
```

```
1
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
5
You have not input any entries.
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
6
a_vitale3@ares:~$ exit
exit

Script done on 2023-12-07 18:00:29-06:00 [COMMAND_EXIT_CODE="0"]
```