

Phone Book

User Manual and Project Description

Lara Hossam 6853
Nour Hesham 7150
Ali Mekky 6850
Seif Tahtawy 6825
Mohamed Alaa 6886

January 29, 2021



Introduction

Digital PhoneBook is a simple, user friendly, phone number, address, email, date of birth, and contacts organizing software. This program is designed to be fast and user friendly. It has many customizable features for quick and simple searching. You can add a new contact, modify an old one, or even delete an entire batch of contacts. This program is most beneficial for business contacts and big phonebooks.

Contents

1	Project Description	3
1.1	Main Functions	4
1.2	Checking the input's validity	5
1.2.1	Validity of email	5
1.2.2	Validity of phone number	5
1.2.3	Validity of date of birth	6
1.2.4	Upper case and lower case in names	6
1.3	EXTRA FEATURE: Actions recap	7
2	Main Algorithms	8
2.1	Search Algorithm	8
2.2	Sort Algorithm	9
2.2.1	Sort by Date of Birth	9
2.2.2	Sort by Last name	9
3	User Manual, How to use the program	10
3.1	User Guide	10
3.1.1	Load the directory	10
3.1.2	Search for a contact	10
3.1.3	Add a new contact	11
3.1.4	Delete an existing contact	11
3.1.5	Modify an existing contact	11
3.1.6	Print the entire contact list (Sorted by Last name)	11
3.1.7	Print the entire contact list (Sorted by Date of birth)	11
3.1.8	Save the changes you have made	12
3.1.9	Quitting the program	12
4	Sample Run of the program	12
5	Source Code	22

1 Project Description

Digital Phone Book's main aim is to make everything easier while dealing with your contacts. User must be able to manipulate his contacts by executing one of the following commands:

- **LOAD:** User is prompted the name of file in which the directory of contcats is saved.
- **QUERY (Search):** The system should process a request by the user to look up information about a specific entry.
- **ADD :** The system should prompt the user field by field for the data of a single record and add it to the directory.
- **DELETE:** The user should be prompted for the name – last and first and all records associated with that name should be deleted from the directory.
- **MODIFY:** The user should be prompted for a last name. Then the user should be prompted field by field to modify the information for one of the records (selected by user) that have that last name.
- **Print (Sort):** Print the entire directory, in sorted order. User should choose whether he wants it sorted by last name or date of birth.
- **SAVE:** Save the directory by writing it out to an external file in a format similar to original file.

All the contacts are saved in a text file "PhoneBook.txt", that the user must specify to the system for it to be loaded, each contact is visible in the following format: *last name, first name, date of birth, street address, phone number and email*

When the user is done with all his modifications, he can save all the changes in a new text file "PhoneBookModified.txt" present in the same directory as the last one, and, with the same text formatting.

In this C program, two global structs have been used, in order to be visible to all the functions in the program. We don't need to pass the structure to any function separately.

The structs are:

```
typedef struct{
int day;
int month;
int year;
}date;

typedef struct{
char last_name[30];
char first_name[30];
date DOB;
char Street_address[100];
char phone_number[12];
char email[100];
}contact;
```

1.1 Main Functions

This program's main goal is to offer a user friendly experience, that's why a menu is present, that prompts the user for a choice by means of an input between 1-8, that's how his command will be executed.

The menu offers the following options:

1. SEARCH for a contact.
2. ADD a new contact.
3. DELETE an existing contact.
4. MODIFY data of an existing contact.
5. PRINT the entire contact list (Sorted by Last Name)
6. PRINT the entire contact list (Sorted by Date of birth)
7. SAVE the changes you have made.
8. QUIT the program.

Besides the main functions, this program uses many more functions, each with a separate and special use. Here are their prototypes:

```
void load_phonebook(void);
int display_instructions(void);
int valid_email(char []);
int valid_phone(char []);
int valid_date(int,int,int);
void casesensitive(char word []);
void search_phonebook();
void modify_contact();
void add_contact();
void delete_contact();
void sort_by_name();
void sort_by_dob();
void print();
void print_contact(contact);
void save_contacts();
```

1.2 Checking the input's validity

1.2.1 Validity of email

To make sure an email entered by user is valid, it must be of the following format: *text@domain.com*. This function returns 0 if the email is invalid and then the user is prompted to try again, and returns 1 if the email is valid.

This function:

- Checks if the first letter of the email is a character, because, if it were not the case, the email would surely be invalid.
- If the first character is valid, a loop locates where the dot and "@" symbol exists in the string and stores their indexes in variables at and dot.
- If the email address does not contain a dot or the "@" symbol, then it's indeed invalid, else, the function checks that the "@" precedes the dot by at least two characters (the domain must be at least 2 characters long), then, it's valid.

```
int valid_email(char email[]){
    if(!isalpha(email[0]))
    {
        return 0;
    }
    int at=-1,dot=-1,i;
    for(i=0;i<strlen(email);i++) {
        if(email[i]=='@')
            at = i;
        else if(email[i]=='.')
            dot = i;
    }
    if(at==-1 || dot==-1)
        return 0;
    if(at - dot > -2)
        return 0;
    return !(dot >=strlen(email)-1);
}
```

1.2.2 Validity of phone number

To make sure the phone number entered by user is valid, it must be of the following format: *Starts with 012 or 011 or 010 or 015, and consists of 11 numbers only.*

This function returns 0 if the phone number is invalid and then the user is prompted to try again, and returns 1 if the phone number is valid.

This function:

- Stores the length of the phone number as it's stored in an array of chars in the variable L.
- Using logical operators, if any of the following happens, the phone number will be invalid.
 - If the length of the phone number is not equal to 11

- If the first number is not 0
- If the second number is not 1
- If the third number is not either 2 or 1 or 0 or 5;

```
int valid_phone(char phone[]){
    int L = strlen(phone);
    if((L!=11) || (phone[0]!='0') || (phone[1]!='1')
    || ((phone[2]!='0') && (phone[2]!='1') &&
    (phone[2]!='2')&&(phone[2]!='5'))
        return 0;
    else return 1;
}
```

1.2.3 Validity of date of birth

To make sure the date of birth entered by user is valid, we must take the following into consideration: Year, months and day range, Leap Year, Months that have 30 days, Months that have 31 days.

This functions returns 0 if the date of birth is invalid and then the user is prompted to try again, and returns 1 if the date of birth is valid.

This function:

```
int valid_date(int day,int month,int year) {
    if((month>12 || year>2020) ||
    ((month==1||month==3||month==5||month==7||
    month==8||month==10||month==12) && (day>31))
    || ((month==4 || month==6 || month==9 ||
    month==11) && (day>30)) || ((month==2) &&
    ((year%4==0 && year%100==0 && year%400!=0)
    || (year%4!=0) ) && day>28) || ((month==2)
    && ((year%4==0 && year%100!=0) ||
    (year%4==0 && year%100==0 && year%400==0))
    && (day>29)))
        return 0;
    else return 1;
}
```

1.2.4 Upper case and lower case in names

Whenever the user inputs a first name and a last name that do not follow the standard rules of capitalization (First letter uppercase and after that all lowercase), our function transforms the names into their standard form.

```
void casesensitive(char word[])
{
    word[0] = toupper(word[0]);
    int i;
    for(i = 1; i < strlen(word); i++)
```

```

    {
        if(word[i-1] != ' ') word[i] = tolower(word[i]);
        else if(word[i-1]== ' ')
        {
            word[i] = toupper(word[i]);
        }
    }
}

```

1.3 EXTRA FEATURE: Actions recap

For our program to be as user friendly as possible, we don't assume that the user has memorized all the changes he has made, this is why, before quitting, every single action the user has taken, is displayed on the screen, if the last action he took wasn't to save the file, he is asked if he surely wants to quit or no, this is when the user gets another chance to save his changes.

IMPORTANT EXAMPLE: If the user wants to add a new contact, save his changes, then visualize his directory sorted by last name and he doesn't want to save it this way, he chooses to quit the program, **OUR RECAP FEATURE** assures him he only saved the new contact, and he didn't sort his contacts, without this function, he would've only received a warning that his changes weren't saved, he'd be prompted to save again, and have a non desired format of his contacts!

Here's how it's done!

A global array of structs is created to hold in each element, an array of the change made, all ordered following the program execution:

```

typedef struct{
    char change[100];
}recap;
recap info[100];
int j=0;

```

At the end of each function, the element `change[j++]` is filled with the exact change made, here's an example after modifying a contact:

We used the `sprintf()` function that formats and stores a series of characters and values in the array buffer.

```

char buffer[100];
int o = sprintf(buffer, 100, "You modified
an existing contact %s %s.", list[x].first_name,
list[x].last_name);
strcpy(info[j++].change, buffer);

```

2 Main Algorithms

2.1 Search Algorithm

In this program , we used *LINEAR SEARCH*.

SEARCH is not case sensitive, if you enter mohamed instead of Mohamed, search will still be successful.

You can exit the function anytime and return to main menu if you click "0", which is a feature that makes of our program a more user friendly application.

If during the linear search process, the requested item is not found, an error message is visible and the user is prompted to try again.

```
void search_phonebook(){
    puts("\tNote: Searching is not case sensitive.
    Inputs \"Mohamed\" and \"mohamed\" are
    considered the same.");
    int i;
    char temp[20];
    while(1)
    {
        int flag = 0;
        puts("Enter the last name you want to search
        for,\t or press zero to exit:");
        scanf("%s", temp);
        if(strcmp(temp,"0") == 0)
        {
            break;
        }
        puts("\tSearching...");
        for(i = 0; i < count; i++)
        {
            if(strcasecmp(list[i].last_name, temp) == 0)
            {
                printf("\tContact Found #%d:",flag+1);
                print_contact(list[i]);
                flag++;
            }
        }
        if(flag == 0)
        {
            puts("No contacts found. Try again!");
        }
        else
        {
            break;
        }
    }
}
```



```

    }
}

```

2.2 Sort Algorithm

In this program , we used *BUBBLE SORT*. Bubble sort which is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

Concerning the sorting by name function, it simply sorts all the contacts by Last name, but in case two or more contacts have the same last name, the bubble sort will go on again to sort the first names of those people, this being done, our phone book will be alphabetically sorted following last name and first name.

2.2.1 Sort by Date of Birth

```

void sort_by_dob(){
    contact temp;
    int i,pass,sorted=0;
    for(pass=1;pass<count && !sorted;pass++){
        sorted = 1;
        for(i=0;i<count-pass;i++){
            if(list[i].DOB.year>list[i+1].DOB.year ||
               (list[i].DOB.year==list[i+1].DOB.year &&
                list[i].DOB.month>list[i+1].DOB.month) ||
               (list[i].DOB.year==list[i+1].DOB.year &&
                list[i].DOB.month==list[i+1].DOB.month &&
                list[i].DOB.day>list[i+1].DOB.day)){
                temp=list[i];
                list[i]=list[i+1];
                list[i+1]=temp;
                sorted = 0;
            }
        }
    }
    print();
}

```

2.2.2 Sort by Last name

```

void sort_by_name(){
    int i,pass,sorted=0;
    contact temp;
    for(pass=1;pass<count && !sorted;pass++)
    {
        sorted = 1;
    }
}

```

```

        for(i=0;i<count-pass;i++)
        {
if(strcmp(list[i].last_name,list[i+1].last_name)>)
        {
                temp=list[i];
                list[i]=list[i+1];
                list[i+1]=temp;
                sorted = 0;
        }
        }
    }
    sorted=0;

    for(pass=1;pass<count && !sorted;pass++)
    {
        sorted = 1;
        for(i=0;i<count-pass;i++)
        {
if((strcmp(list[i].last_name,list[i+1].last_name)
==0)&&(strcmp(list[i].first_name,list[i+1].
first_name)>0))
        {
                temp=list[i];
                list[i]=list[i+1];
                list[i+1]=temp;
                sorted = 0;
        }
        }
    }

    print();
    save=0;
}

```

3 User Manual, How to use the program

3.1 User Guide

3.1.1 Load the directory

1. Write down the name of the text file in which all your contacts are saved.

3.1.2 Search for a contact

1. Choose number 1 from the display menu.
2. Enter the Last name of the contact you wish to search for.
3. If you don't wish to continue, you can choose 0 and return back to menu.

4. After your search is done successfully, Press any key to get back to menu.

3.1.3 Add a new contact

1. Choose number 2 from the display menu.
2. Enter the data of your new contact field by field. Make sure to enter them in the correct format, otherwise, you'll be prompted to try again.
3. Press any key to get back to menu.

3.1.4 Delete an existing contact

1. Choose number 3 from the display menu.
2. Enter the Last and First name of the contact you wish to delete. Take care: You can't delete a contact that's not in your directory, if you enter invalid data, you'll be asked to try again.
3. Press any key to get back to menu.

3.1.5 Modify an existing contact

1. Choose number 4 from the display menu.
2. Enter the last name of the contact you'd like to modify.
3. Choose the number of the record that contains the same last name by entering its number shown on the screen.
4. Modify the data of this contact.
5. If you don't wish to continue, you can choose 0 and return back to menu.
6. After the modification is successfully done, Press any key to get back to menu.

3.1.6 Print the entire contact list (Sorted by Last name)

1. Choose number 5 from the display menu.
2. Press any key to get back to menu.

3.1.7 Print the entire contact list (Sorted by Date of birth)

1. Choose number 6 from the display menu.
2. Press any key to get back to menu.

3.1.8 Save the changes you have made

1. Choose number 7 from the display menu.
2. Press any key to get back to menu.
3. You can access the data in a text file created in the same folder, named "PhoneBook-Modified.txt"

3.1.9 Quitting the program

Your are show a full detailed list of all the changes made during the program's execution. If you have not saved your last change, the program will notify you and will ask you if you're sure you want to quit.

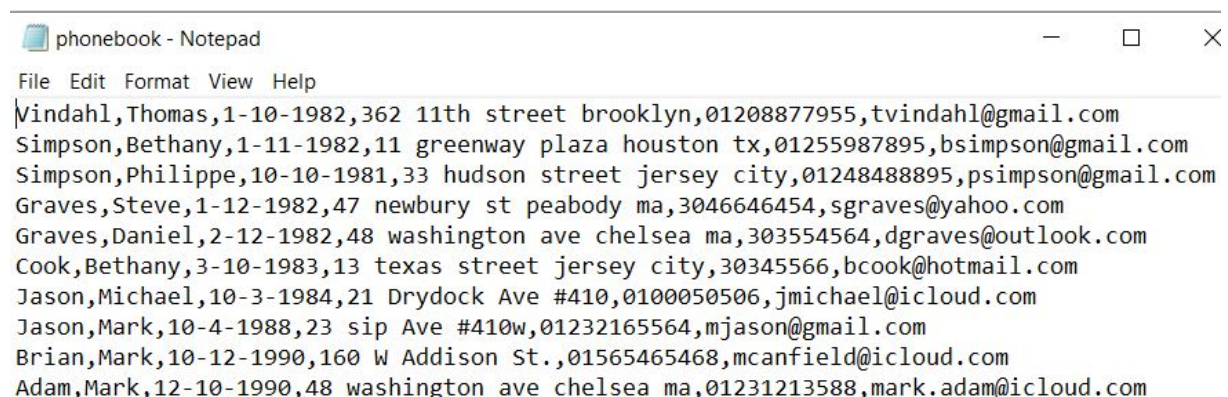
If you have saved your changes, the program quits successfully.

4 Sample Run of the program

Welcome to our sample run!

In this sample run, we will load the contacts from a file named "PhoneBook.txt", we will search for a contact named Bethany Simpson, we will add a contact named Claire Beauchamp, modify the contact Bethany Cook, Print the entire phone book directory sorted by last name then save our changes,we will delete a contact Thomas Vindahl,then we will print the entire phone book sorted by date of birth and save again our changes. Finally,we will delete contact named Mark Adam. We will try to quit without saving our last change, then the warning will remind us that the last change isn't saved, we'll go back and choose the save option, then quit. The saved changes are present in a a text file named PhoneBook-Modified.txt.

This is the file loaded before modification.



```
phonebook - Notepad
File Edit Format View Help
Vindahl,Thomas,1-10-1982,362 11th street brooklyn,01208877955,tvindahl@gmail.com
Simpson,Bethany,1-11-1982,11 greenway plaza houston tx,01255987895,bsimpson@gmail.com
Simpson,Philippe,10-10-1981,33 hudson street jersey city,01248488895,psimpson@gmail.com
Graves,Steve,1-12-1982,47 newbury st peabody ma,3046646454,sgraves@yahoo.com
Graves,Daniel,2-12-1982,48 washington ave chelsea ma,303554564,dgraves@outlook.com
Cook,Bethany,3-10-1983,13 texas street jersey city,30345566,bcook@hotmail.com
Jason,Michael,10-3-1984,21 Drydock Ave #410,0100050506,jmichael@icloud.com
Jason,Mark,10-4-1988,23 sip Ave #410w,01232165564,mjason@gmail.com
Brian,Mark,10-12-1990,160 W Addison St.,01565465468,mcanfield@icloud.com
Adam,Mark,12-10-1990,48 washington ave chelsea ma,01231213588,mark.adam@icloud.com
```

```
WELCOME TO YOUR PHONE BOOK
Your best way to deal with your contacts!
Please enter the name of the file you wish to open: phonebook.txt
```

```
WELCOME TO YOUR PHONE BOOK
Your best way to deal with your contacts!
Please enter the name of the file you wish to open: phonebook.txt
File loaded successfully.

Please choose an option:
1- SEARCH for a contact.
2- ADD a new contact.
3- DELETE an existing contact.
4- MODIFY data of an existing contact.
5- PRINT the entire contact list (Sorted by Last Name)
6- PRINT the entire contact list (Sorted by Date of birth)
7- SAVE the changes you have made.
8- QUIT the program.
1
```

```
Note: Searching is not case sensitive. Inputs "Mohamed" and "mohamed" are considered the same.
Enter the last name you want to search for, or press zero to exit:
simpson
```

```
Note: Searching is not case sensitive. Inputs "Mohamed" and "mohamed" are considered the same.
Enter the last name you want to search for, or press zero to exit:
simpson
Searching...
Contact Found #1:
Last Name Simpson
First Name Bethany
Date of birth 1-11-1982
Street address 11 greenway plaza houston tx
Phone number 01255987895
Email address bsimpson@gmail.com

Contact Found #2:
Last Name Simpson
First Name Philippe
Date of birth 10-10-1981
Street address 33 hudson street jersey city
Phone number 01248488895
Email address psimpson@gmail.com

Press any key to get back to menu.
```

```

Please choose an option:
1-    SEARCH for a contact.
2-    ADD a new contact.
3-    DELETE an existing contact.
4-    MODIFY data of an existing contact.
5-    PRINT the entire contact list (Sorted by Last Name)
6-    PRINT the entire contact list (Sorted by Date of birth)
7-    SAVE the changes you have made.
8-    QUIT the program.

```

2

```

Enter contact's last name: Beauchamp
Enter contact's first name: claire
Enter contact's date of birth DD/MM/YYYY: 2 81 1988
INVALID INPUT. TRY AGAIN.
Re-enter contact's date of birth DD/MM/YYYY: 2 8 1988
Enter contact's street address: Inverness, Scotland
Enter contact's phone number: 01279477941
Enter contact's email: claire_healer@yahoo.com
CONTACT ADDED SUCCESSFULLY!
Press any key to get back to menu.

```

```

Please choose an option:
1-    SEARCH for a contact.
2-    ADD a new contact.
3-    DELETE an existing contact.
4-    MODIFY data of an existing contact.
5-    PRINT the entire contact list (Sorted by Last Name)
6-    PRINT the entire contact list (Sorted by Date of birth)
7-    SAVE the changes you have made.
8-    QUIT the program.

```

4

```

Note: Searching is not case sensitive. Inputs "Mohamed" and "mohamed" are considered the same.
Enter last name of the contact you would like to modify, or press zero to exit:
cook
    Searching...
    Contact Found #1:
Last Name      Cook
First Name     Bethany
Date of birth  3-10-1983
Street address 13 texas street jersey city
Phone number   30345566
Email address  bcCook@hotmail.com

```

```

Enter # of contact to modify, or press zero to exit:

```

1

```
Enter # of contact to modify, or press zero to exit:  
1
```

```
You are now modifying the contact "Cook, Bethany"
```

```
Enter new last name:
```

```
cook
```

```
Enter new first name:
```

```
Brianna
```

```
Enter new street address:
```

```
Rue Charles de Gaule, Paris
```

```
Enter new phone number:
```

```
010016732711
```

```
INVALID INPUT. TRY AGAIN.
```

```
Re-enter new phone number:
```

```
01001373271
```

```
Enter new date of birth (DD/MM/YYYY):
```

```
25 2 2001
```

```
Enter new email address:
```

```
brianna.harvard@gmail.com
```

```
DATA MODIFIED SUCCESSFULLY!
```

```
Press any key to get back to menu.
```

```
Please choose an option:
```

- 1- SEARCH for a contact.
- 2- ADD a new contact.
- 3- DELETE an existing contact.
- 4- MODIFY data of an existing contact.
- 5- PRINT the entire contact list (Sorted by Last Name)
- 6- PRINT the entire contact list (Sorted by Date of birth)
- 7- SAVE the changes you have made.
- 8- QUIT the program.

```
5
```

```
#1 Contact
Last Name      Adam
First Name     Mark
Date of birth   12-10-1990
Street address 48 washington ave chelsea ma
Phone number    01231213588
Email address   mark.adam@icloud.com

#2 Contact
Last Name      Beauchamp
First Name     Claire
Date of birth   2-8-1988
Street address  Inverness, Scotland
Phone number    01279477941
Email address   claire_healer@yahoo.com

#3 Contact
Last Name      Brian
First Name     Mark
Date of birth   10-12-1990
Street address  160 W Addison St.
Phone number    01565465468
Email address   mcanfield@icloud.com

#4 Contact
Last Name      Cook
First Name     Brianna
Date of birth   25-2-2001
Street address  Rue Charles de Gaule, Paris
Phone number    01001373271
Email address   brianna_harvard@gmail.com

#5 Contact
Last Name      Graves
First Name     Daniel
Date of birth   2-12-1982
Street address  48 washington ave chelsea ma
Phone number    303554564
Email address   dgraves@outlook.com
```


#6 Contact
Last Name Graves
First Name Steve
Date of birth 1-12-1982
Street address 47 newbury st peabody ma
Phone number 3046646454
Email address sgraves@yahoo.com

#7 Contact
Last Name Jason
First Name Mark
Date of birth 10-4-1988
Street address 23 sip Ave #410w
Phone number 01232165564
Email address mjason@gmail.com

#8 Contact
Last Name Jason
First Name Michael
Date of birth 10-3-1984
Street address 21 Drydock Ave #410
Phone number 0100050506
Email address jmichael@icloud.com

#9 Contact
Last Name Simpson
First Name Bethany
Date of birth 1-11-1982
Street address 11 greenway plaza houston tx
Phone number 01255987895
Email address bsimpson@gmail.com

#10 Contact
Last Name Simpson
First Name Philippe
Date of birth 10-10-1981
Street address 33 hudson street jersey city
Phone number 01248488895
Email address psimpson@gmail.com

#11 Contact
Last Name Vindahl
First Name Thomas
Date of birth 1-10-1982
Street address 362 11th street brooklyn
Phone number 01208877955
Email address tvindahl@gmail.com

Press any key to get back to menu.

Please choose an option:

- 1- SEARCH for a contact.
- 2- ADD a new contact.
- 3- DELETE an existing contact.
- 4- MODIFY data of an existing contact.
- 5- PRINT the entire contact list (Sorted by Last Name)
- 6- PRINT the entire contact list (Sorted by Date of birth)
- 7- SAVE the changes you have made.
- 8- QUIT the program.

7

DATA SAVED SUCCESSFULLY!

Press any key to get back to menu.

Please choose an option:

- 1- SEARCH for a contact.
- 2- ADD a new contact.
- 3- DELETE an existing contact.
- 4- MODIFY data of an existing contact.
- 5- PRINT the entire contact list (Sorted by Last Name)
- 6- PRINT the entire contact list (Sorted by Date of birth)
- 7- SAVE the changes you have made.
- 8- QUIT the program.

3

Note: Deleting is not case sensitive. Inputs "Mohamed" and "mohamed" are considered the same.

Enter the first name for the contact you want to delete, or press zero to exit:

thomas

Enter the last name for the contact you want to delete, or press zero to exit:

vindahl

Contact successfully deleted

Press any key to get back to menu.

Please choose an option:

- 1- SEARCH for a contact.
- 2- ADD a new contact.
- 3- DELETE an existing contact.
- 4- MODIFY data of an existing contact.
- 5- PRINT the entire contact list (Sorted by Last Name)
- 6- PRINT the entire contact list (Sorted by Date of birth)
- 7- SAVE the changes you have made.
- 8- QUIT the program.

6

```
#1 Contact
Last Name      Simpson
First Name     Philippe
Date of birth  10-10-1981
Street address 33 hudson street jersey city
Phone number   01248488895
Email address  psimpson@gmail.com

#2 Contact
Last Name      Simpson
First Name     Bethany
Date of birth  1-11-1982
Street address 11 greenway plaza houston tx
Phone number   01255987895
Email address  bsimpson@gmail.com

#3 Contact
Last Name      Graves
First Name     Steve
Date of birth  1-12-1982
Street address 47 newbury st peabody ma
Phone number   3046646454
Email address  sgraves@yahoo.com

#4 Contact
Last Name      Graves
First Name     Daniel
Date of birth  2-12-1982
Street address 48 washington ave chelsea ma
Phone number   303554564
Email address  dgraves@outlook.com

#5 Contact
Last Name      Jason
First Name     Michael
Date of birth  10-3-1984
Street address 21 Drydock Ave #410
Phone number   0100050506
Email address  jmichael@icloud.com
```

```
#6 Contact
Last Name      Jason
First Name     Mark
Date of birth   10-4-1988
Street address 23 sip Ave #410w
Phone number    01232165564
Email address   mjason@gmail.com
```

```
#7 Contact
Last Name      Beauchamp
First Name     Claire
Date of birth   2-8-1988
Street address  Inverness, Scotland
Phone number    01279477941
Email address   claire_healer@yahoo.com
```

```
#8 Contact
Last Name      Adam
First Name     Mark
Date of birth   12-10-1990
Street address  48 washington ave chelsea ma
Phone number    01231213588
Email address   mark.adam@icloud.com
```

```
#9 Contact
Last Name      Brian
First Name     Mark
Date of birth   10-12-1990
Street address  160 W Addison St.
Phone number    01565465468
Email address   mcanfield@icloud.com
```

```
#10 Contact
Last Name      Cook
First Name     Brianna
Date of birth   25-2-2001
Street address  Rue Charles de Gaule, Paris
Phone number    01001373271
Email address   brianna_harvard@gmail.com
```

Press any key to get back to menu.

Please choose an option:

- 1- SEARCH for a contact.
- 2- ADD a new contact.
- 3- DELETE an existing contact.
- 4- MODIFY data of an existing contact.
- 5- PRINT the entire contact list (Sorted by Last Name)
- 6- PRINT the entire contact list (Sorted by Date of birth)
- 7- SAVE the changes you have made.
- 8- QUIT the program.

7

DATA SAVED SUCCESSFULLY!

Press any key to get back to menu.

```

Please choose an option:
1-    SEARCH for a contact.
2-    ADD a new contact.
3-    DELETE an existing contact.
4-    MODIFY data of an existing contact.
5-    PRINT the entire contact list (Sorted by Last Name)
6-    PRINT the entire contact list (Sorted by Date of birth)
7-    SAVE the changes you have made.
8-    QUIT the program.

```

3

```

Note: Deleting is not case sensitive. Inputs "Mohamed" and "mohamed" are considered the same.
Enter the first name for the contact you want to delete, or press zero to exit:
mark
Enter the last name for the contact you want to delete, or press zero to exit:
adam
Contact successfully deleted
Press any key to get back to menu.

```

```

Please choose an option:
1-    SEARCH for a contact.
2-    ADD a new contact.
3-    DELETE an existing contact.
4-    MODIFY data of an existing contact.
5-    PRINT the entire contact list (Sorted by Last Name)
6-    PRINT the entire contact list (Sorted by Date of birth)
7-    SAVE the changes you have made.
8-    QUIT the program.

```

8

```

Here is a recap of what you have done:
1    You searched for contacts with last name Simpson.
2    You added a new contact Claire Beauchamp.
3    You modified an existing contact Brianna Cook.
4    You sorted your contacts by Last Name.
5    You saved your previous change.
6    You deleted an existing contact Thomas Vindahl.
7    You sorted your contacts by Date of Birth.
8    You saved your previous change.
9    You deleted an existing contact Mark Adam.

```

```

Your last change hasn't been saved, are you sure you want to exit? (Y/N)
If you want to exit anyway input 'Y', if not, input 'N' to get back to menu.
N

```

```

Please choose an option:
1-    SEARCH for a contact.
2-    ADD a new contact.
3-    DELETE an existing contact.
4-    MODIFY data of an existing contact.
5-    PRINT the entire contact list (Sorted by Last Name)
6-    PRINT the entire contact list (Sorted by Date of birth)
7-    SAVE the changes you have made.
8-    QUIT the program.

```

7


```
DATA SAVED SUCCESSFULLY!  
Press any key to get back to menu.
```

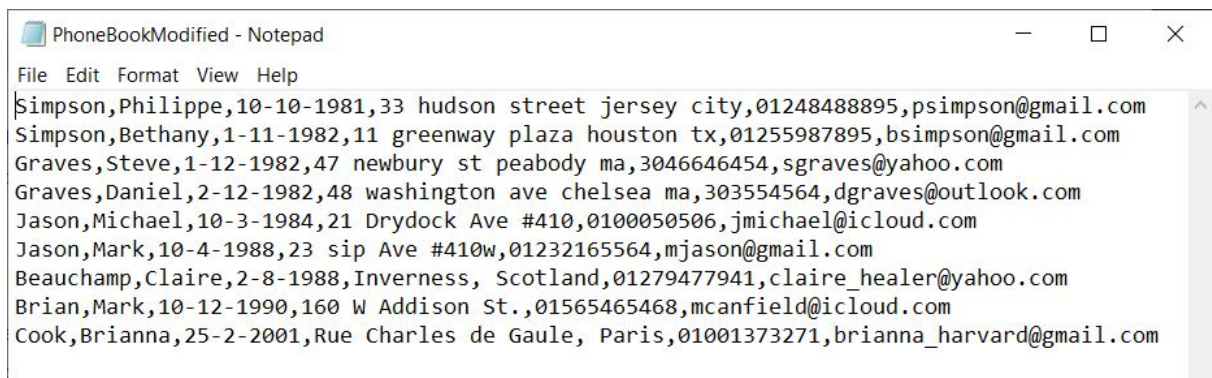
```
Please choose an option:  
1- SEARCH for a contact.  
2- ADD a new contact.  
3- DELETE an existing contact.  
4- MODIFY data of an existing contact.  
5- PRINT the entire contact list (Sorted by Last Name)  
6- PRINT the entire contact list (Sorted by Date of birth)  
7- SAVE the changes you have made.  
8- QUIT the program.  
8
```

```
Here is a recap of what you have done:  
1 You searched for contacts with last name Simpson.  
2 You added a new contact Claire Beauchamp.  
3 You modified an existing contact Brianna Cook.  
4 You sorted your contacts by Last Name.  
5 You saved your previous change.  
6 You deleted an existing contact Thomas Vindahl.  
7 You sorted your contacts by Date of Birth.  
8 You saved your previous change.  
9 You deleted an existing contact Mark Adam.  
10 You saved your previous change.
```

```
You are now quitting your phonebook.  
We really hope it was useful!  
Goodbye!
```

```
Process returned 0 (0x0) execution time : 512.634 s  
Press any key to continue.
```

Modified text file is:



```
File Edit Format View Help  
Simpson,Philippe,10-10-1981,33 hudson street jersey city,01248488895,psimpson@gmail.com  
Simpson,Bethany,1-11-1982,11 greenway plaza houston tx,01255987895,bsimpson@gmail.com  
Graves,Steve,1-12-1982,47 newbury st peabody ma,3046646454,sgraves@yahoo.com  
Graves,Daniel,2-12-1982,48 washington ave chelsea ma,303554564,dgraves@outlook.com  
Jason,Michael,10-3-1984,21 Drydock Ave #410,010050506,jmichael@icloud.com  
Jason,Mark,10-4-1988,23 sip Ave #410w,01232165564,mjason@gmail.com  
Beauchamp,Claire,2-8-1988,Inverness, Scotland,01279477941,claire_healer@yahoo.com  
Brian,Mark,10-12-1990,160 W Addison St.,01565465468,mcanfield@icloud.com  
Cook,Brianna,25-2-2001,Rue Charles de Gaule, Paris,01001373271,brianna_harvard@gmail.com
```

5 Source Code

```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 #define QUIT 8  
4 #include <string.h>  
5 #include <strings.h>  
6 #include <conio.h>  
7 #include <windows.h>  
8 #include <ctype.h>
```

```

9
10 int count=0;
11 char junk;
12 int save=1;
13 int flagDelete=0;
14 int j=0;
15
16 typedef struct{
17 char change[100];
18 }recap;
19
20 typedef struct{
21 int day;
22 int month;
23 int year;
24 }date;
25
26 typedef struct{
27 char last_name[30];
28 char first_name[30];
29 date DOB;
30 char Street_address[100];
31 char phone_number[12];
32 char email[100];
33 }contact;
34
35 contact list[100];
36 recap info[100];
37
38
39 void print();
40 int display_instructions();
41 void load_phonebook();
42 int valid_email(char email[]);
43 int valid_phone(char phone[]);
44 int valid_date(int day, int month, int year);
45 void print_contact(contact person);
46 void search_phonebook();
47 void add_contact();
48 void delete_contact();
49 void modify_contact();
50 void sort_by_name();
51 void sort_by_dob();
52 void save_contacts();
53 void casesensitive(char word[]);
54
55 int display_instructions(void){
56     int choice;
57     puts("Please choose an option:");
58     puts("\t1 -\tSEARCH for a contact.");
59     puts("\t2 -\tADD a new contact.");
60     puts("\t3 -\tDELETE an existing contact.");
61     puts("\t4 -\tMODIFY data of an existing contact.");
62     puts("\t5 -\tPRINT the entire contact list (Sorted by Last
63     Name)");
64     puts("\t6 -\tPRINT the entire contact list (Sorted by Date of
65     birth)");
66     puts("\t7 -\tSAVE the changes you have made.");

```

```

65     puts("\t8-\tQUIT the program.");
66     scanf("%d",&choice);
67
68     system("cls");
69
70     return choice;
71 }
72 void load_phonebook(void){
73     char name[50];
74     FILE* fp;
75     printf("Please enter the name of the file you wish to open: ");
76     gets(name);
77     if((fp=fopen(name,"r"))==NULL)
78     {
79         fprintf(stderr,"ERROR IN LOADING FILE.");
80         exit(1);
81     }
82     else printf("File loaded successfully.\n\n\n");
83     while(!feof(fp))
84     {
85         fscanf(fp,"\n%[^,]",list[count].last_name);
86         fscanf(fp,"%[^,]",list[count].first_name);
87         fscanf(fp,"%d-%d-%d",&list[count].DOB.day,&list[count].
DOB.month,&list[count].DOB.year);
88         fscanf(fp,"%[^,]",list[count].Street_address);
89         fscanf(fp,"%[^,]",list[count].phone_number);
90         fscanf(fp,"%[^\\n]",list[count].email);
91         count++;
92     }
93     fclose(fp);
94 }
95 int main(){
96     puts("\t\t\tWELCOME TO YOUR PHONE BOOK");
97     puts("\t\t\tYour best way to deal with your contacts!");
98     int choice=0;
99     int choice_2=0;
100    int flag=0;
101    int times=0,generic;
102    load_phonebook();
103    while(flag==0)
104    {
105        if(times>0)
106        {
107            printf("Press any key to get back to menu.\n");
108            getch();
109            system("cls");
110        }
111        choice=display_instructions();
112        switch (choice)
113        {
114            case 1: {search_phonebook(); break;}
115            case 2: {add_contact(); break;}
116            case 3: {delete_contact(); break;}
117            case 4: {modify_contact(); break;}
118            case 5: {sort_by_name(); break;}
119            case 6: {sort_by_dob(); break;}
120            case 7: {save_contacts(); break;}

```



```

121         case QUIT:
122         {
123
124             if (save == 1)
125             {
126                 puts("Here is a recap of what you have done:"
127 );
128                 for (j=0; j<times; j++)
129                     printf("\t%d\t%s\n", j+1, info[j].change);
130
131                 printf("\n\tYou are now quitting your
132 phonebook.\n\tWe really hope it was useful!\n\tGoodbye!\n");
133                 exit(0);
134             }
135             else if (save == 0)
136             {
137                 puts("Here is a recap of what you have done:"
138 );
139                 for (j=0; j<times; j++)
140                     printf("\t%d\t%s\n", j+1, info[j].change);
141
142                 puts("\nYour last change hasn't been saved,
143 are you sure you want to exit? (Y/N)");
144                 char letter[2];
145                 while (1)
146                 {
147                     puts("If you want to exit anyway input 'Y
148 ', if not, input 'N' to get back to menu.");
149                     scanf("%s", letter);
150                     if ((letter[0] == 'Y') || (letter[0] == 'y'))
151                     exit(0);
152                     else if ((letter[0] == 'N') || (letter[0] == '
153 n'))
154                     {
155                         times = times - 1;
156                         break;
157                     }
158                     else puts("Please enter a valid input.");
159                 }
160             }
161             break;
162         }
163         default:
164         {
165             puts("Please choose a number between 1-8. Try again."
166 );
167             times = -1;
168             break;
169         }
170     }
171
172     times++;
173
174 }
175
176 return 0;

```

```

171 }
172 void modify_contact() {
173     puts("\tNote: Searching is not case sensitive. Inputs \"
Mohamed\" and \"mohamed\" are considered the same.");
174     int i, x;
175     char temp[30];
176     while (1)
177     {
178         puts("Enter last name of the contact you would like to
modify,\tor press zero to exit:");
179         scanf("%s", temp);
180         if(strcmp(temp, "0") == 0)
181         {
182             break;
183         }
184         puts("\tSearching ... ");
185         int arr[count];
186         int flag = 0;
187         for(i = 0; i < count; i++)
188         {
189             if(strcasecmp(list[i].last_name, temp) == 0)
190             {
191                 printf("\tContact Found #%d:", flag+1);
192                 arr[flag] = i;
193                 print_contact(list[i]);
194                 flag++;
195             }
196         }
197         if(flag == 0)
198         {
199             puts("No contacts found. Try again!");
200         }
201         else
202         {
203             int number;
204             puts("Enter # of contact to modify,\tor press zero to
exit:");
205             scanf("%d", &number);
206             if(number == 0)
207             {
208                 break;
209             }
210             int flag2=0;
211             for(i = 0; i < flag; i++)
212             {
213                 if(arr[number] == arr[i+1])
214                 {
215                     x = arr[i];
216                     printf("\nYou are now modifying the contact
\"%s, %s\\\"\\n", list[x].last_name, list[x].first_name);
217                     puts("Enter new last name:");
218                     scanf("%s", list[x].last_name);
219                     casesensitive(list[x].last_name);
220                     puts("Enter new first name:");
221                     scanf("%s", list[x].first_name);
222                     casesensitive(list[x].first_name);
223                     fflush(stdin);
224

```

```

225         puts("Enter new street address:");
226         scanf("%s", list[x].Street_address);
227         fflush(stdin);
228         puts("Enter new phone number:");
229         gets(list[x].phone_number);
230         while(!valid_phone(list[x].phone_number))
231         {
232             puts("INVALID INPUT. TRY AGAIN.");
233             puts("Re-enter new phone number:");
234             scanf("%s", list[x].phone_number);
235         }
236         fflush(stdin);
237         puts("Enter new date of birth (DD/MM/YYYY):");
238         ;
239         scanf("%d/%d/%d", &list[x].DOB.day, &junk,
240         &list[x].DOB.month, &junk, &list[x].DOB.year);
241         while(!valid_date(list[x].DOB.day, list[x].DOB
242         .month, list[x].DOB.year))
243         {
244             puts("INVALID INPUT. TRY AGAIN.");
245             puts("Re-enter new date of birth:");
246             scanf("%d/%d/%d", &list[x].DOB.day, &
247             junk, &list[x].DOB.month, &junk, &list[x].DOB.year);
248         }
249         fflush(stdin);
250         puts("Enter new email address:");
251         scanf("%s", list[x].email);
252         while(!valid_email(list[x].email))
253         {
254             puts("INVALID INPUT. TRY AGAIN.");
255             puts("Re-enter new email address:");
256             scanf("%s", list[x].email);
257         }
258         flag2 = 1;
259     }
260     }
261     if(!flag2)
262     {
263         puts("Error occurred. Wrong choice. Try again!");
264     }
265     else if(flag2)
266     {
267         puts("DATA MODIFIED SUCCESSFULLY!");
268         break;
269     }
270 }
271 }
272 save=0;
273 char buffer[100];
274 int o = snprintf(buffer, 100, "You modified an existing
275 contact %s %s.", list[x].first_name, list[x].last_name);
276 strcpy(info[j++].change, buffer);
277 }
278 void sort_by_name() {
279     int i, pass, sorted=0;
280     contact temp;
281     for (pass=1; pass<count && !sorted; pass++)

```

```

278 {
279     sorted = 1;
280     for (i=0; i<count-pass; i++)
281     {
282         if (strcmp(list[i].last_name, list[i+1].last_name) > 0)
283         {
284             temp = list[i];
285             list[i] = list[i+1];
286             list[i+1] = temp;
287             sorted = 0;
288         }
289     }
290 }
291 sorted = 0;
292
293 for (pass = 1; pass < count && !sorted; pass++)
294 {
295     sorted = 1;
296     for (i=0; i<count-pass; i++)
297     {
298         if ((strcmp(list[i].last_name, list[i+1].last_name) == 0) && (
299             strcmp(list[i].first_name, list[i+1].first_name) > 0))
300         {
301             temp = list[i];
302             list[i] = list[i+1];
303             list[i+1] = temp;
304             sorted = 0;
305         }
306     }
307
308     print();
309     save = 0;
310     strcpy(info[j++].change, "You sorted your contacts by Last
311     Name.");
312 }
313 void sort_by_dob() {
314     contact temp;
315     int i, pass, sorted = 0;
316     for (pass = 1; pass < count && !sorted; pass++) {
317         sorted = 1;
318         for (i = 0; i < count - pass; i++) {
319             if (list[i].DOB.year > list[i+1].DOB.year || (list[i].
320                 DOB.year == list[i+1].DOB.year && list[i].DOB.month > list[i+1].
321                 DOB.month) || (list[i].DOB.year == list[i+1].DOB.year && list[i].
322                 DOB.month == list[i+1].DOB.month && list[i].DOB.day > list[i+1].
323                 DOB.day)) {
324                 temp = list[i];
325                 list[i] = list[i+1];
326                 list[i+1] = temp;
327                 sorted = 0;
328             }
329         }
330     }
331     print();
332     save = 0;

```

```

330         strcpy(info[j++].change, "You sorted your contacts by
Date of Birth.");
331
332
333     }
334     void print() {
335         int i;
336         for (i=0; i<count; i++){
337             printf("#%d Contact\n", i+1);
338             printf("Last Name \t %s\n", list[i].last_name);
339             printf("First Name \t %s\n", list[i].first_name);
340             printf("Date of birth \t %d-%d-%d\n", list[i].DOB.day, list
[i].DOB.month, list[i].DOB.year);
341             printf("Street address \t %s\n", list[i].Street_address);
342             printf("Phone number \t %s\n", list[i].phone_number);
343             printf("Email address \t %s\n", list[i].email);
344             printf("\n\n");
345         }
346         save=0;
347     }
348 }
349 void print_contact(contact person){
350     printf("\n");
351     printf("Last Name \t %s\n", person.last_name);
352     printf("First Name \t %s\n", person.first_name);
353     printf("Date of birth \t %d-%d-%d\n", person.DOB.day,
person.DOB.month, person.DOB.year);
354     printf("Street address \t %s\n", person.Street_address);
355     printf("Phone number \t %s\n", person.phone_number);
356     printf("Email address \t %s\n", person.email);
357     printf("\n\n");
358     save=0;
359 }
360 }
361 void add_contact() {
362     fflush(stdin);
363     printf("Enter contact's last name: ");
364     scanf("%s", list[count].last_name);
365     casesensitive(list[count].last_name);
366     fflush(stdin);
367
368     printf("Enter contact's first name: ");
369     scanf("%s", list[count].first_name);
370     casesensitive(list[count].first_name);
371     fflush(stdin);
372     printf("Enter contact's date of birth DD/MM/YYYY: ");
373     scanf("%d/%d/%d", &list[count].DOB.day, &junk, &list[count].
DOB.month, &junk, &list[count].DOB.year);
374     while (valid_date(list[count].DOB.day, list[count].DOB.month,
list[count].DOB.year) == 0) {
375         puts("INVALID INPUT. TRY AGAIN.");
376         printf("Re-enter contact's date of birth DD/MM/YYYY: ");
377         scanf("%d/%d/%d", &list[count].DOB.day, &junk, &list[
count].DOB.month, &junk, &list[count].DOB.year);
378     }
379     fflush(stdin);
380     printf("Enter contact's street address: ");
381     scanf("%s", list[count].Street_address);

```

```

382     fflush(stdin);
383     printf("Enter contact's phone number: ");
384     scanf("%s", list[count].phone_number);
385     while(valid_phone(list[count].phone_number) ==0) {
386         puts("INVALID INPUT. TRY AGAIN.");
387         printf("Re-enter contact's phone number: ");
388         scanf("%s", list[count].phone_number);
389     }
390     printf("Enter contact's email: ");
391     scanf("%s", list[count].email);
392     while(valid_email(list[count].email) == 0) {
393         puts("INVALID INPUT. TRY AGAIN.");
394         printf("Re-enter contact's email: ");
395         scanf("%s", list[count].email);
396     }
397     char buffer[100];
398     int o = snprintf(buffer, 100, "You added a new contact %s %s.",
399     list[count].first_name, list[count].last_name);
400     strcpy(info[j++].change, buffer);
401     count++;
402     printf("CONTACT ADDED SUCCESSFULLY!\n");
403     save=0;
404 }
405 }
406 int valid_date(int day, int month, int year) {
407     if((month>12 || year>2020) ||
408     ((month==1||month==3||month==5||month==7||month==8||month
409     ==10||month==12) && (day>31))
410     || ((month==4 || month==6 || month==9 || month==11) && (day
411     >30))
412     || ((month==2) && ((year%4==0 && year%100==0 && year%400!=0)
413     || (year%4!=0) ) && day>28)
414     || ((month==2) && ((year%4==0 && year%100!=0) || (year%4==0
415     && year%100==0 && year%400==0)) && (day>29)))
416     return 0;
417     else return 1;
418 }
419 int valid_phone(char phone[]) {
420     int L = strlen(phone);
421     if((L!=11) || (phone[0]!='0') || (phone[1]!='1') || ((phone
422     [2]!='0') && (phone[2]!='1') && (phone[2]!='2')&&(phone[2]!='5
423     '))))
424     return 0;
425     else return 1;
426 }
427 int valid_email(char email[]) {
428     if(!isalpha(email[0]))
429     {
430         return 0;
431     }
432     int at=-1, dot=-1, i;
433     for(i=0; i<strlen(email); i++) {
434         if(email[i]=='@')
435             at = i;
436         else if(email[i]=='.')
437             dot = i;

```

```

443     }
444     if (at == -1 || dot == -1)
445         return 0;
446     if (at - dot > -2)
447         return 0;
448     return !(dot >= strlen(email) - 1);
449 }
450 void save_contacts(void) {
451     FILE* p;
452     if ((p = fopen("PhoneBookModified.txt", "w")) == NULL)
453     {
454         fprintf(stderr, "ERROR IN SAVING FILE.\n");
455     }
456     else
457     {
458         for (int i = 0; i < count; i++) {
459             fprintf(p, "%s, ", list[i].last_name);
460             fprintf(p, "%s, ", list[i].first_name);
461             fprintf(p, "%d-%d-%d, ", list[i].DOB.day, list[i].DOB.month,
462 list[i].DOB.year);
463             fprintf(p, "%s, ", list[i].Street_address);
464             fprintf(p, "%s, ", list[i].phone_number);
465             fprintf(p, "%s\n", list[i].email);
466         }
467         fclose(p);
468         printf("DATA SAVED SUCCESSFULLY!\n");
469         save = 1;
470     }
471     char buffer[100];
472     int o = snprintf(buffer, 100, "You saved your previous change
473 .");
474     strcpy(info[j++].change, buffer);
475 }
476 void search_phonebook() {
477     puts("\tNote: Searching is not case sensitive. Inputs \"
478 Mohamed\" and \"mohamed\" are considered the same.");
479     int i, x;
480     char temp[20];
481     while(1)
482     {
483         int flag = 0;
484         puts("Enter the last name you want to search for, \t or
485 press zero to exit:");
486         scanf("%s", temp);
487         if(strcmp(temp, "0") == 0)
488         {
489             break;
490         }
491         puts("\tSearching ...");
492         for(i = 0; i < count; i++)
493         {
494             if(strcasecmp(list[i].last_name, temp) == 0)
495             {
496                 printf("\tContact Found #d:", flag+1);
497                 print_contact(list[i]);
498                 x = i;

```

```

487         flag++;
488     }
489 }
490 }
491 if(flag == 0)
492 {
493     puts("No contacts found. Try again!");
494 }
495 }
496 else
497 {
498     break;
499 }
500 }
501 }
502 save=1;
503 char buffer[100];
504 int o = snprintf(buffer, 100, "You searched for contacts
with last name %s.", list[x].last_name);
505 strcpy(info[j++].change, buffer);
506 }
507 }
508 void delete_contact() {
509     fflush(stdin);
510     contact temp;
511     char first[20], last[20];
512     int i, k, x;
513     puts("\tNote: Deleting is not case sensitive. Inputs \"
Mohamed\" and \"mohamed\" are considered the same.");
514     printf("Enter the first name for the contact you want to
delete, \t or press zero to exit:\n");
515     scanf("%s", first);
516     if(strcmp(first, "0")== 0)
517     {
518         return;
519     }
520     fflush(stdin);
521     printf("Enter the last name for the contact you want to
delete, \t or press zero to exit:\n");
522     scanf("%s", last);
523     if(strcmp(last, "0")== 0)
524     {
525         return;
526     }
527     fflush(stdin);
528     for(i=0; i<count; i++)
529     {
530         if(strcasecmp(list[i].first_name, first)==0 && strcmp(list[i].last_name, last)==0)
531         {
532             x=i;
533             char buffer[100];
534             int o = snprintf(buffer, 100, "You deleted an existing
contact %s %s.", list[x].first_name, list[x].last_name);
535             strcpy(info[j++].change, buffer);
536             for(k=i; k<count; k++){
537                 temp=list[k];
538                 list[k]=list[k+1];

```



```

539         list[k+1]=temp;
540     }
541     flagDelete++;
542
543
544     }else
545     continue;
546     }
547     if(flagDelete)
548     {
549         printf("Contact successfully deleted\n");
550         count=count-1;
551     }
552     else
553     {
554         printf("Contact not found, please try again!\n\n\n\n");
555         delete_contact();
556     }
557     save=0;
558
559 }
560
561 void casesensitive(char word[])
562 {
563     word[0] = toupper(word[0]);
564     int i;
565     for(i = 1; i < strlen(word); i++)
566     {
567         if(word[i-1] != ' ') word[i] = tolower(word[i]);
568         else if(word[i-1]== ' ')
569         {
570             word[i] = toupper(word[i]);
571         }
572     }
573 }
574 }

```