

PHONE DIRECTORY

C programming project

Prepared by

Amr Yasser Imam

6772

Hazem Mohamed Abdallah

6723

Table of contents

Section no.	Content			#
1	Introduction			3
2	Overview to c	1	General info	4
		2	Relation to other languages	5
		3	History of C	5
3	Program description	1	Project plan	7
		2	Main purpose and future scope	13
4	Program design			13
5	Algorithms	1	Search algorithm	14
		2	Sorting algorithm	15
6	Source code			16
7	Sample runs			36
8	User manual	1	Installed commands	40
		2	First step	41
		3	Commands' operation way	42
		4	Fields of errors	48
9	References			49



1 Introduction

We have discussed so far various features of C language and are ready to write and execute program of modest complexity. However, before attempting to develop complex programs, it is worthwhile to consider some programming techniques that would help design efficient and error free program.

The program development process includes three important stages, namely, program design, program coding and program testing. All the three stages contribute to the production of high-quality program.

In “**PHONE DIRECTORY**” we have done system design, source coding, and program testing and added many more features to facilitate the use of the program. User can load data of contacts from file, add new contacts, delete certain contacts, modify certain contacts, save modifications and many more options.

This program will decrease human errors when using typical hard directories which will give high security, data consistency, easy handling, easy data editing, easy data keeping, and many more benefits.



2 Overview to C

2.1 General Info

C is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system. By design, C provides constructs that map efficiently to typical machine instructions. It has found lasting use in applications previously coded in assembly language. Such applications include operating systems and various application software for computer architectures that range from supercomputers to PLCs and embedded systems.

A successor to the programming language *B*, C was originally developed at Bell Labs by Dennis Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages, with C compilers from various vendors available for the majority of existing computer architectures and operating systems. C has been standardized by the ANSI since 1989 (ANSI C) and by the International Organization for Standardization (ISO). As of January 2021, C is the most popular programming language.

C is an imperative procedural language. It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind



can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

2.2 Relation to other languages

Many later languages have borrowed directly or indirectly from C, including C++, C#, Unix's C shell, D, Go, Java, JavaScript (including trans pilers), Julia, Limbo, LPC, Objective-C, Perl, PHP, Python, Ruby, Rust, Swift, Verilog and System Verilog (hardware description languages). These languages have drawn many of their control structures and other basic features from C. Most of them (Python being a dramatic exception) also express highly similar syntax to C, and they tend to combine the recognizable expression and statement syntax of C with underlying type systems, data models, and semantics that can be radically different.

2.3 History of C

The origin of C is closely tied to the development of the Unix operating system, originally implemented in assembly language on a PDP-7 by Dennis Ritchie and Ken Thompson, incorporating several ideas from colleagues. Eventually, they decided to port the operating system to a PDP-11. The original PDP-11 version of Unix was also developed in assembly language.

Thompson desired a programming language to make utilities for the new platform. At first, he tried to make a Fortran compiler, but soon gave up the idea. Instead, he created a cut-down version of the recently developed BCPL systems programming language. The official description of BCPL was not available at the time, and Thompson modified the syntax to be less wordy, producing the similar but somewhat simpler B. However, few utilities were



ultimately written in B because it was too slow, and B could not take advantage of PDP-11 features such as byte addressability.

In 1972, Ritchie started to improve B, which resulted in creating a new language C. The C compiler and some utilities made with it were included in Version 2 Unix.

At Version 4 Unix, released in November 1973, the Unix kernel was extensively re-implemented in C. By this time, the C language had acquired some powerful features such as struct types.

Preprocessor was introduced around 1973 at the urging of Alan Snyder and also in recognition of the usefulness of the file-inclusion mechanisms available in BCPL and PL/I. Its original version provided only included files and simple string replacements: `#include` and `#define` of parameter less macros. Soon after that, it was extended, mostly by Mike Lesk and then by John Reiser, to incorporate macros with arguments and conditional compilation.

Unix was one of the first operating system kernels implemented in a language other than assembly. Earlier instances include the Multics system (which was written in PL/I) and Master Control Program (MCP) for the Burroughs B5000 (which was written in ALGOL) in 1961. In around 1977, Ritchie and Stephen C. Johnson made further changes to the language to facilitate portability of the Unix operating system. Johnson's Portable C Compiler served as the basis for several implementations of C on new platforms.



3 Program description

3.1 Project plan

Phase (1) – Program’s main functions and preparations:

- Define struct of date (birth day, birth month, birth year).
- Define struct of contacts (Fname^{char}, Lname^{char}, DOB^{int, int, int}, Phone number^{char}, Address^{char}, E-mail address^{char}, Age^{int}).
- * Phone no.’s type is char to avoid crashes if it contains letters whether when user type it or read from a file.
- Define some global variables such as array of contacts “c[200]”, count of contacts, currently opened file name.... etc.
- Load
 - Get file name from user.
 - Check file’s existence, if exists, open it with “r” mode and store file name, otherwise, ask user to enter a correct file name.
 - Scan the data of each contact in the file to the array of contacts known that data in file is comma delimited. [^,],.
 - Close file.
- Query
 - Get a name from the user.
 - Search for given name with 2-step search (exactly same name, substring).
 - Print results data (1 result or >1 result print data, 0 print not found).
- Add
 - Get number of new contacts from the user “n”.



- Increase the count of contacts with “n”.
- Ask user to fill the data of each new contact field by field and it to the array.
- Delete
 - Get full name of user.
 - Search for contacts with given full name with 2-step search (exactly same name, substring).
 - Print results (1 result or >1 result print data, 0 print not found).
 - Ask whether to delete results or not (y/n question).
 - If >1 result ask user which one to delete.
 - If (yes) make contacts’ indices just after chosen contact decrease by 1.
- Modify
 - Get a name from user.
 - Search for given name with 2-step search (exactly same name, substring).
 - Print results data (1 result or >1 result print data, 0 print not found).
 - Ask whether to modify results or not (y/n question).
 - If >1 result ask user which one to modify.
 - If (yes) ask user to fill modified data of chosen contact field by field.
- Print
 - Ask user which type of sorting to use (by dob “ascendingly” – by Lname).
 - If user choose sorting by DOB then sort first by Lname then by DOB (better look).



- Sort directory according to user choice.
- Print the entire directory in a table form.
- Save
 - Ask user whether to save in same file opened or not (y/n question).
 - If (yes) overwrite data in the same file with comma delimitation.
 - If (no) ask user for file name then write data in the new file name with comma delimitation.
- Quit
 - Check if data saved or not.
 - If not, warn user that it is not save and ask him/her whether to save it or not.
 - If saved, exit program.

Phase (2) – Supporting functions:

- E-mail errors:
 - Check that e-mail contains exactly 1 “@” and contains no spaces.
 - Prefix:
 - 1- Validate that the prefix’s length does not exceed 64 characters and not equal 0.
 - 2- Validate that first character is one of (a-z) or (0-9) characters.
 - 3- Validate that the entire prefix does not contain two consecutive periods, dashes or underscores.
 - 4- Validate that the entire prefix consists only of (a-z), (0-9), periods, dashes and underscores.
 - 5- Validate that last character is one of (a-z) or (0-9) characters.



- Domain:

- 1- Validate that the domain's length does not exceed 253 character.
- 2- Validate that first character is one of (a-z) or (0-9) characters.
- 3- Validate that domain contain at least one period.
- 4- Validate that the entire domain does not contain two consecutive periods or dashes.
- 5- Validate that the entire domain consists only of (a-z), (0-9), periods and dashes.
- 6- Validate that the last period is followed by the top-level domain.

- Errors of a contact:

- DOB:

- 1- Validate that birth day is between 1 and 31.
- 2- Validate that birth month is between 1 and 12.
- 3- Validate that birth year is between 1903 and 2021.

- Phone number:

- 1- Validate that the number does not contain any (a-z) characters.

- E-mail:

In case of finding any errors provide user with full name of contact and field of error (dob, number or e-mail) then ask him/her to enter the correct data.

- Correct letters case:

- Function to change string case of a name to be alphabetically correct.

- Substring:

- Function that get names and check if they are part of other names or not, in order to provide more results upon any search.

- Check answers of y/n questions:



- Function to validate answers of y/n questions whether user answered by (yes, y, no, n) or another answer.
- Sorting contacts:
 - By DOB (ascendingly).
 - By Lname (a to z).
- Ask for sorting:
 - Function to ask user which type of sorting to use.

Phase (3) – Operation function and UI:

- Operation function (commands operators):
 - Function gets commands from user and execute it.
 - Basic commands:
 - 1- LOAD.
 - 2- QUERY.
 - 3- ADD.
 - 4- MODIFY.
 - 5- DELETE.
 - 6- PRINT.
 - 7- SAVE.
 - 8- QUIT.
 - Function operates till user quit program.
- User interface (UI):
 - Program must provide a menu of available commands for the user upon starting the program.
 - User will see (Type a command >>), then he/she give the program a certain command and program executes it.



Phase (4) – Extras:

- Following are some options planned to be added:

- Autosave:

Program saves everything done by user automatically in the same file.

- Print currently opened file name.
- Sort file.

Phase (5) – Enhancing and testing:

- Enhance printing look.
- Decrease number of lines if possible.
- Test a real directory.

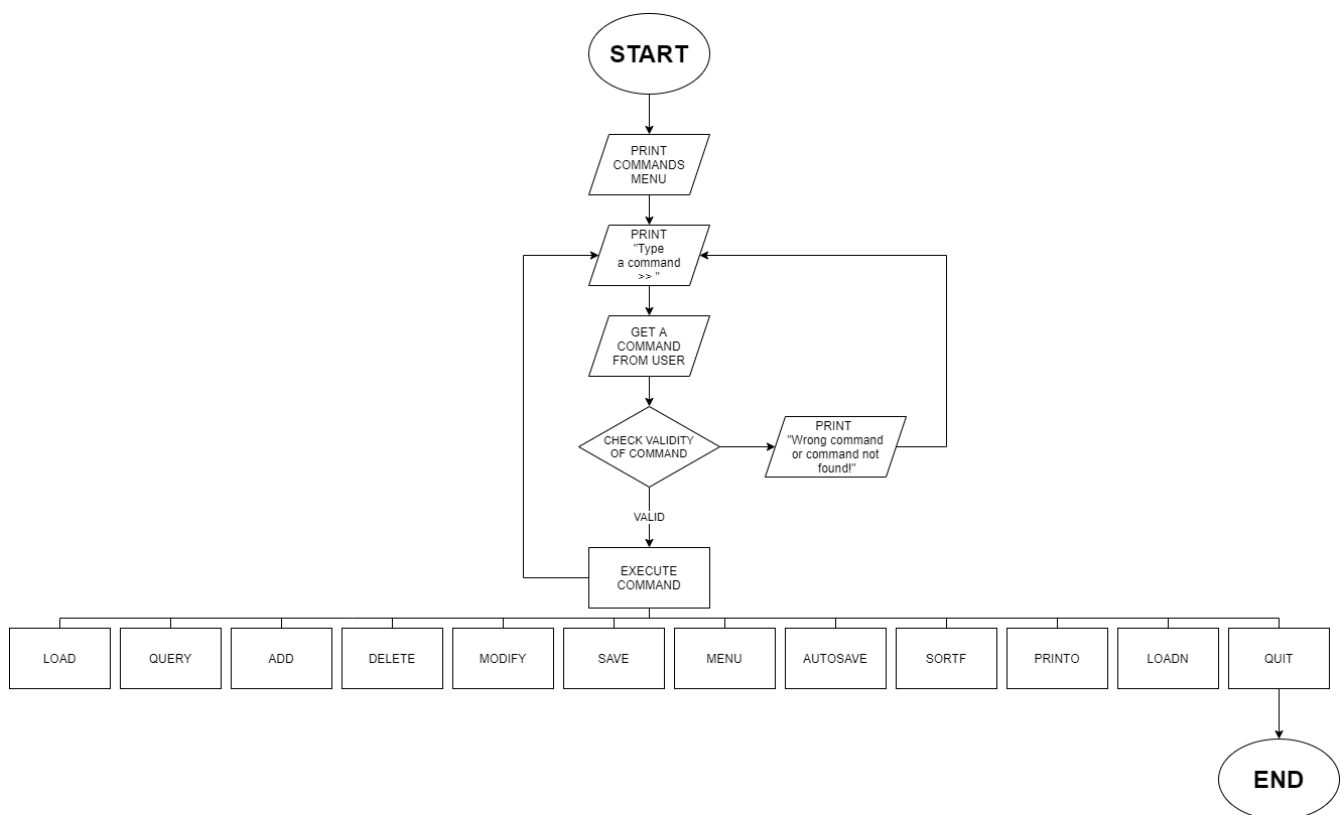


3.2 Main purpose and future scope

Program main purpose is to get rid of using hard phone directories. In hard phone directories we may face a lot of problems or we may lose it completely

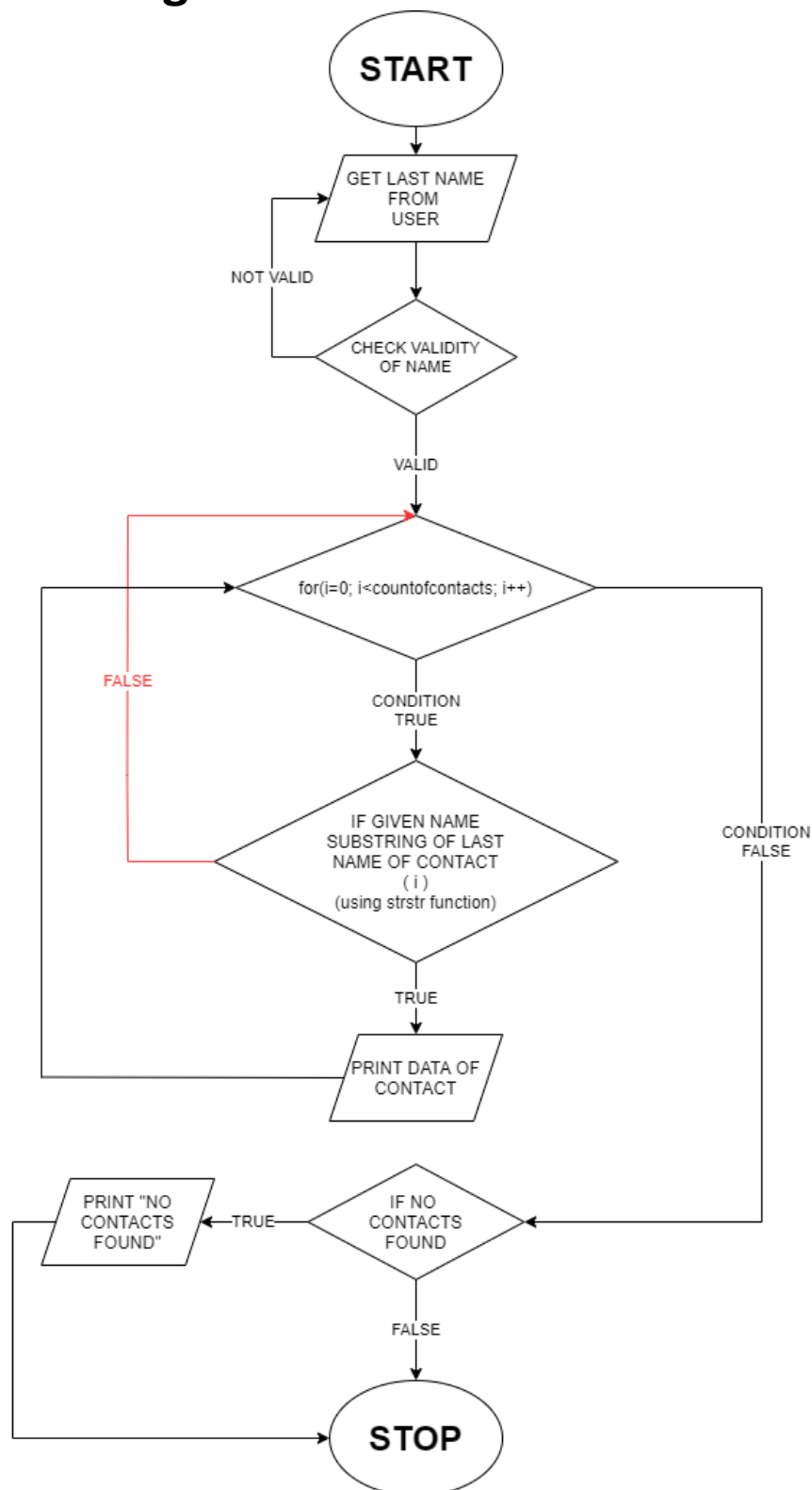
In future, we are planning to add saving in excel sheet, adding hidden keys and adding option to save data as a table.

4 Program design

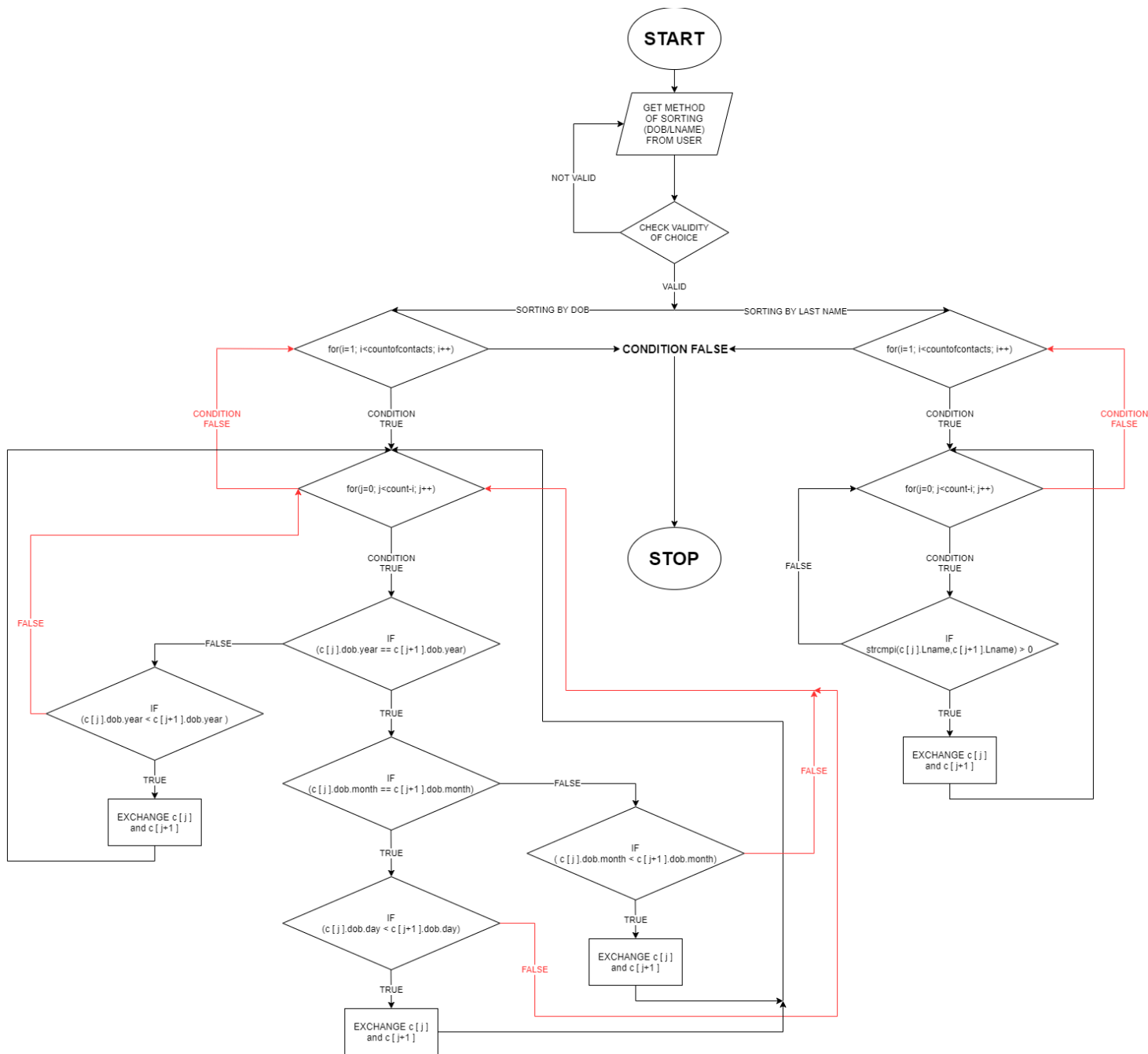


5 Algorithms

5.1 Search algorithm



5.2 Sorting algorithm



6 Source code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

typedef struct
{
    int day;
    int month;
    int year;
} DOB; //contact dob

typedef struct
{
    char Fname[35];
    char Lname[35];
    char address[100];
    char number[15];
    char email[100];
    int age;
    DOB dob;
} contact; //one contact data

contact c[200]; //struct array of 200 contacts
int count=0,load=0,autosave=0,saved=0,edit=0;
char OrigFileName[25]; //stores opened file name for any further use

int EmailErrors(char s[])
{
    int i,x=0,periodcounter=0,Atcounter=0,index[15];
    char temp[strlen(s)],*tok;
    for(i=0; i<strlen(s); i++) //@ checker
    {
        if( s[i]=='@')
            Atcounter++;
    }
    if (Atcounter != 1) return 1;
    for(i=0; i<strlen(s); i++) //space checker
    {
        if( s[i]==' ')
            return 2;
    }
    strcpy(temp,s);
    strtok(temp,"@");
    //prefix
    //checking length
    if ((strlen(temp) > 64)) return 3;
    //checking first char
    if ((temp[0] != 'a' && (temp[0] != 'b') && (temp[0] != 'c') && (temp[0] != 'd') && (temp[0] != 'e') && (temp[0] != 'f') &&
(temp[0] != 'g') && (temp[0] != 'h') && (temp[0] != 'i') && (temp[0] != 'j')
    && (temp[0] != 'k') && (temp[0] != 'l') && (temp[0] != 'm') && (temp[0] != 'n') && (temp[0] != 'o') && (temp[0] !=
'p') && (temp[0] != 'q') && (temp[0] != 'r') && (temp[0] != 's') && (temp[0] != 't'))
```




```

    && (temp[0] != 'u') && (temp[0] != 'v') && (temp[0] != 'w') && (temp[0] != 'x') && (temp[0] != 'y') && (temp[0] !=
'z') && (temp[0] != '0') && (temp[0] != '1') && (temp[0] != '2') && (temp[0] != '3')
    && (temp[0] != '4') && (temp[0] != '5') && (temp[0] != '6') && (temp[0] != '7') && (temp[0] != '8') && (temp[0] !=
'9'))
    return 4;
    for (i=1; i<strlen(temp); i++)
    {
        //checking for 2 consecutive periods, dashes or underscores
        if((temp[i] == '.') || (temp[i] == '-') || (temp[i] == '_'))
        {
            if((temp[i+1] == '.') || (temp[i+1] == '-') || (temp[i+1] == '_'))
                return 5;
        }
        //checking entire prefix
        else if ((temp[i] != 'a') && (temp[i] != 'b') && (temp[i] != 'c') && (temp[i] != 'd') && (temp[i] != 'e') && (temp[i] != 'f')
&& (temp[i] != 'g') && (temp[i] != 'h') && (temp[i] != 'i') && (temp[i] != 'j')
&& (temp[i] != 'k') && (temp[i] != 'l') && (temp[i] != 'm') && (temp[i] != 'n') && (temp[i] != 'o') && (temp[i] != 'p')
&& (temp[i] != 'q') && (temp[i] != 'r') && (temp[i] != 's') && (temp[i] != 't')
&& (temp[i] != 'u') && (temp[i] != 'v') && (temp[i] != 'w') && (temp[i] != 'x') && (temp[i] != 'y') && (temp[i] != 'z')
&& (temp[i] != '0') && (temp[i] != '1') && (temp[i] != '2') && (temp[i] != '3')
&& (temp[i] != '4') && (temp[i] != '5') && (temp[i] != '6') && (temp[i] != '7') && (temp[i] != '8') && (temp[i] != '9')
&& (temp[i] != '_') && (temp[i] != '.') && (temp[i] != '-')) return 6;
    }
    //checking last char
    if ((temp[strlen(temp)-1] != 'a') && (temp[strlen(temp)-1] != 'b') && (temp[strlen(temp)-1] != 'c') &&
(temp[strlen(temp)-1] != 'd') && (temp[strlen(temp)-1] != 'e') && (temp[strlen(temp)-1] != 'f') && (temp[strlen(temp)-1]
!= 'g') && (temp[strlen(temp)-1] != 'h') && (temp[strlen(temp)-1] != 'i') && (temp[strlen(temp)-1] != 'j')
&& (temp[strlen(temp)-1] != 'k') && (temp[strlen(temp)-1] != 'l') && (temp[strlen(temp)-1] != 'm') &&
(temp[strlen(temp)-1] != 'n') && (temp[strlen(temp)-1] != 'o') && (temp[strlen(temp)-1] != 'p') && (temp[strlen(temp)-1]
!= 'q') && (temp[strlen(temp)-1] != 'r') && (temp[strlen(temp)-1] != 's') && (temp[strlen(temp)-1] != 't')
&& (temp[strlen(temp)-1] != 'u') && (temp[strlen(temp)-1] != 'v') && (temp[strlen(temp)-1] != 'w') &&
(temp[strlen(temp)-1] != 'x') && (temp[strlen(temp)-1] != 'y') && (temp[strlen(temp)-1] != 'z') && (temp[strlen(temp)-1]
!= '0') && (temp[strlen(temp)-1] != '1') && (temp[strlen(temp)-1] != '2') && (temp[strlen(temp)-1] != '3')
&& (temp[strlen(temp)-1] != '4') && (temp[strlen(temp)-1] != '5') && (temp[strlen(temp)-1] != '6') &&
(temp[strlen(temp)-1] != '7') && (temp[strlen(temp)-1] != '8') && (temp[strlen(temp)-1] != '9')) return 7;
    strcpy(temp,s);
    tok=strtok(temp,"@");
    tok=strtok(NULL,"");
    strcpy(temp,tok);
    //domain
    //checking length
    if ((strlen(temp) > 253)) return 8;
    //checking first char
    if ((temp[0] != 'a') && (temp[0] != 'b') && (temp[0] != 'c') && (temp[0] != 'd') && (temp[0] != 'e') && (temp[0] != 'f') &&
(temp[0] != 'g') && (temp[0] != 'h') && (temp[0] != 'i') && (temp[0] != 'j')
&& (temp[0] != 'k') && (temp[0] != 'l') && (temp[0] != 'm') && (temp[0] != 'n') && (temp[0] != 'o') && (temp[0] !=
'p') && (temp[0] != 'q') && (temp[0] != 'r') && (temp[0] != 's') && (temp[0] != 't')
&& (temp[0] != 'u') && (temp[0] != 'v') && (temp[0] != 'w') && (temp[0] != 'x') && (temp[0] != 'y') && (temp[0] !=
'z') && (temp[0] != '0') && (temp[0] != '1') && (temp[0] != '2') && (temp[0] != '3')
&& (temp[0] != '4') && (temp[0] != '5') && (temp[0] != '6') && (temp[0] != '7') && (temp[0] != '8') && (temp[0] !=
'9'))
    return 9;
    for(i=0; i<strlen(temp); i++)
    {
        //checking that domain contains at least one period

```



```

    if ((temp[i]=='.'))
    {
        periodcounter++;
        index[x]=i;
        x++;
    }
}
if(!periodcounter) return 10;
for (i=1; i<strlen(temp); i++)
{
    //checking for 2 consecutive periods, dashes or underscores
    if((temp[i] == '.') || (temp[i] == '-'))
    {
        if((temp[i+1] == '.') || (temp[i+1] == '-'))
            return 11;
    }
    //checking entire domain
    else if ((temp[i] != 'a') && (temp[i] != 'b') && (temp[i] != 'c') && (temp[i] != 'd') && (temp[i] != 'e') && (temp[i] != 'f')
    && (temp[i] != 'g') && (temp[i] != 'h') && (temp[i] != 'i') && (temp[i] != 'j')
    && (temp[i] != 'k') && (temp[i] != 'l') && (temp[i] != 'm') && (temp[i] != 'n') && (temp[i] != 'o') && (temp[i] != 'p')
    && (temp[i] != 'q') && (temp[i] != 'r') && (temp[i] != 's') && (temp[i] != 't')
    && (temp[i] != 'u') && (temp[i] != 'v') && (temp[i] != 'w') && (temp[i] != 'x') && (temp[i] != 'y') && (temp[i] != 'z')
    && (temp[i] != '0') && (temp[i] != '1') && (temp[i] != '2') && (temp[i] != '3')
    && (temp[i] != '4') && (temp[i] != '5') && (temp[i] != '6') && (temp[i] != '7') && (temp[i] != '8') && (temp[i] != '9')
    && (temp[i] != '.') && (temp[i] != '-')) return 12;
}
//checking the top-level domain
if (strlen(temp)>(index[periodcounter-1]+4)) return 13;
if ((temp[index[periodcounter-1]+1] == 'c'))
{
    if ((temp[index[periodcounter-1]+2] != 'o')) return 14;
    else if ((temp[index[periodcounter-1]+3] != 'm')) return 14;
}
else if ((temp[index[periodcounter-1]+1] == 'o'))
{
    if ((temp[index[periodcounter-1]+2] != 'r')) return 14;
    else if ((temp[index[periodcounter-1]+3] != 'g')) return 14;
}
else if ((temp[index[periodcounter-1]+1] == 'n'))
{
    if ((temp[index[periodcounter-1]+2] != 'e')) return 14;
    else if ((temp[index[periodcounter-1]+3] != 't')) return 14;
}
else if ( (temp[index[periodcounter-1]+1] != 'c') && (temp[index[periodcounter-1]+1] != 'o') &&
(temp[index[periodcounter-1]+1] != 'n')) return 14;
return 0;
}

void CorrectLettersCase(char s[]) //function to change letter cases of names
{
    strlwr(s);
    s[0]=toupper(s[0]);
}

void ErrorsOfContact()

```



```

{
    int i,j;
    //DOB
    for(i=0; i<count; i++)
    {
        while (c[i].dob.day < 1 || c[i].dob.day > 31)
        {
            printf("\nContact %s %s with birth day \"%d\", has wrong birth day, birth day must be between 1 and 31.\nEnter
the correct birth day: ",c[i].Fname,c[i].Lname,c[i].dob.day);
            scanf("%d",&c[i].dob.day);
            getchar();
        }
        while (c[i].dob.month < 1 || c[i].dob.month > 12)
        {
            printf("\nContact %s %s with birth month \"%d\", has wrong birth day, birth month must be between 1 and
12.\nEnter the correct birth month: ",c[i].Fname,c[i].Lname,c[i].dob.month);
            scanf("%d",&c[i].dob.month);
            getchar();
        }
        while (c[i].dob.year > 2021 || c[i].dob.year < 1903)
        {
            printf("\nContact %s %s with birth year \"%d\", has wrong birth year, birth year must be between 1903 and
2021.\nEnter the correct birth year: ",c[i].Fname,c[i].Lname,c[i].dob.year);
            scanf("%d",&c[i].dob.year);
            getchar();
        }
    }
    //Number
    for(i=0; i<count; i++)
    {
        while (1)
        {
            for(j=0; j<strlen(c[i].number); j++)
            {
                while(c[i].number[j] != '0' && c[i].number[j] != '1' && c[i].number[j] != '2' && c[i].number[j] != '3' &&
c[i].number[j] != '4' && c[i].number[j] != '5' && c[i].number[j] != '6' && c[i].number[j] != '7' && c[i].number[j] != '8' &&
c[i].number[j] != '9')
                {
                    printf("\nContact %s %s with phone number \"%s\", has wrong number, phone number must contain only
numbers.\nEnter the correct number: ",c[i].Fname,c[i].Lname,c[i].number);
                    gets(c[i].number);
                    j=0;
                }
                if(c[i].number[j]==' ')
                {
                    printf("\nContact %s %s with phone number \"%s\", has wrong number, phone number must not contain
spaces.\nEnter the correct number: ",c[i].Fname,c[i].Lname,c[i].number);
                    gets(c[i].number);
                }
            }
            break;
        }
    }
}
//Email

```



```

int emailerror;
for(i=0; i<count; i++)
{
    emailerror=EmailErrors(c[i].email);
    while(emailerror)
    {
        switch (emailerror)
        {
            case 1:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, email must contain exactly one \"@\".\n\nEnter
the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 2:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, e-mail must not contain spaces.\n\nEnter the
correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 3:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, prefix's length must not exceed 64
characters.\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 4:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, prefix's first character must be an ascii letter (a-z)
or number (0-9).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 5:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, prefix must not contain consecutive periods (.),
dashes (-) or underscores (_).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 6:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, only letters (a-z), numbers(0-9), periods (.),
dashes(-), and underscores ( _) are allowed in the prefix.\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 7:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, prefix's last character must be an ascii letter (a-z)
or number (0-9).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 8:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, domain's length must exceed 253
characters.\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 9:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, domain's first character must be an ascii letter (a-
z) or number (0-9).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 10:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, domain must contains at least one period
(.).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 11:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, domain must not contain consecutive periods (.)
or dashes (-).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
                break;
            case 12:
                printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, only letters (a-z), numbers(0-9), periods (.), and
dashes(-) are allowed in the domain.\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);

```



```

        break;
    case 13:
        printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, top-level domain must be one of (.com - .org -
.net).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
        break;
    case 14:
        printf("\nContact %s %s with e-mail \"%s\", has wrong e-mail, top-level domain must be one of (.com - .org -
.net).\n\nEnter the correct e-mail: ",c[i].Fname,c[i].Lname,c[i].email);
        break;
    }
    if(!emailererror) break;
    gets(c[i].email);
    strlwr(c[i].email);
    emailerror=EmailErrors(c[i].email);
}
}
}

```

```

int CheckAnswerYes_No(char s[]) //function to get error of yes/no questions
{
    if(strlen(s) == 1)
    {
        if(!strcmpi(s,"y"))
            return 1;
        else if (!strcmpi(s,"n"))
            return 0;
        else
        {
            printf("\a\nWrong input, Please enter a correct answer: ");
            gets(s);
            CheckAnswerYes_No(s);
        }
    }
    else
    {
        if (!strcmpi(s,"yes"))
            return 1;
        else if (!strcmpi(s,"no"))
            return 0;
        else
        {
            printf("\a\nWrong input, Please enter a correct answer: ");
            gets(s);
            CheckAnswerYes_No(s);
        }
    }
}

```

```

void ReadContactsFromFile() //constructor function for a single contact in a file
{
    count=0;
    char *Filename = (char *)malloc(25);
    FILE *f;
    printf("\n\nEnter file path (txt files only): ");
    gets(Filename);
}

```



```

f=fopen(Filename,"r");
while(f==NULL) //loop operates while the user enters incorrect path
{
    printf("\n\aFile not found, Please enter the correct path for the file: ");
    gets(Filename);
    f=fopen(Filename,"r");
}
while (!feof(f)) //scan data of a contact and adding it in the struct array
{
    fscanf(f,"%[^,]",c[count].Lname);
    CorrectLettersCase(c[count].Lname);
    fscanf(f,"%[^,]",c[count].Fname);
    CorrectLettersCase(c[count].Fname);
    fscanf(f,"%d-%d-%d",&c[count].dob.day,&c[count].dob.month,&c[count].dob.year);
    fscanf(f,"%[^,]",c[count].address);
    fscanf(f,"%[^,]",c[count].number);
    fscanf(f,"%[^\\n\\n]",c[count].email);
    strlwr(c[count].email);
    count ++;
}
fclose(f);
strcpy(OrigFileName,Filename);
free(Filename);
printf("\nChecking for errors contact by contact:\n");
ErrorsOfContact();
printf("\nNo errors or no errors remaining!\n");
printf("\n\aFile read successfully!\n");
load=1; //variable detecting that a file has been loaded
}

```

```

void SearchDirectory() //function to search a contact in the directory by last name
{
    int i,j,k=1,counter=0;
    char *Lname = (char*)malloc(35),*Lname1 = (char*)malloc(35);
    printf("\nEnter last name of the contact: ");
    gets(Lname);
    while(Lname[0]!='\0')
    {
        printf("\nNo name entered, please enter a name: ");
        gets(Lname);
    }
    CorrectLettersCase(Lname);
    strcpy(Lname1,Lname);
    strlwr(Lname1);
    if (strlen(Lname) == 1)
    {
        for (i=0; i<count; i++)
        {
            for (j=0; j<(strlen(c[i].Lname)) ; j++)
            {
                if ((strstr(c[i].Lname,Lname)) || (strstr(c[i].Lname,Lname1)))
                {
                    if (k==1)
                        printf("\nContacts found :\n");
                    printf("\n#%d\n",k);
                }
            }
        }
    }
}

```



```

        printf("First name: %s\n",c[i].Fname);
        printf("Last name: %s\n",c[i].Lname);
        printf("Address: %s\n",c[i].address);
        printf("E-mail: %s\n",c[i].email);
        printf("Number: %s\n",c[i].number);
        printf("DOB: %d/%d/%d\n\n",c[i].dob.day,c[i].dob.month,c[i].dob.year);
        k++;
        counter ++;
        break;
    }
}
}
}

else
{
    for(i=0; i<count; i++)
    {
        if ((strstr(c[i].Lname,Lname)) || (strstr(c[i].Lname,Lname1)))
        {
            if (k==1)
                printf("\nContacts found :\n");
            printf("\n#%d\n",k);
            printf("First name: %s\n",c[i].Fname);
            printf("Last name: %s\n",c[i].Lname);
            printf("Address: %s\n",c[i].address);
            printf("E-mail: %s\n",c[i].email);
            printf("Number: %s\n",c[i].number);
            printf("DOB: %d/%d/%d\n\n",c[i].dob.day,c[i].dob.month,c[i].dob.year);
            k++;
            counter ++;
        }
    }
}
if(!counter) printf("\a\nNo contacts found.\n");
free(Lname);
free(Lname1);
}

```

void AddContacts() //function to add new contact in the directory but not saved in file

```

{
    int i,n,j=1;
    printf("\nEnter the number of the contacts you want to add: ");
    scanf("%d",&n);
    getchar();
    i=count;
    count+=n;
    for(i; i<count; i++)
    {
        printf("\nEnter the data for contact #%d\n",j);
        printf("First Name: ");
        gets(c[i].Fname);
        CorrectLettersCase(c[i].Fname);
        printf("Last Name: ");
        gets(c[i].Lname);
    }
}

```



```

    CorrectLettersCase(c[i].Lname);
    printf("Address: ");
    gets(c[i].address);
    printf("Email: ");
    gets(c[i].email);
    strlwr(c[i].email);
    printf("Number: ");
    gets(c[i].number);
    printf("DOB (DD MM YEAR): ");
    scanf("%d %d %d",&c[i].dob.day,&c[i].dob.month,&c[i].dob.year);
    getchar();
    j++;
}
if (n==1) printf("\nChecking for errors of new contact:\n");
else printf("\nChecking for errors of new contacts contact by contact:\n");
ErrorsOfContact();
printf("\nNo errors or no errors remaining!\n");
if(autosave) SaveDirectoryInFile(2);
edit=1;
if(n==1) printf("\n\aaContact added successfully!\n");
else printf("\n\aaContacts added successfully!\n");
}

void DeleteContact() //function to delete a contact in the directory
{
    int i,x,j=1,counter=0,index[count];
    char *Lname = (char *)malloc(35),*Fname = (char *)malloc(35),*answer;
    printf("\nEnter first name of the contact: ");
    gets(Fname);
    while(Fname[0]!='\0')
    {
        printf("\nNo name entered, please enter a name: ");
        gets(Fname);
    }
    printf("\nEnter last name of the contact: ");
    gets(Lname);
    while(Lname[0]!='\0')
    {
        printf("\nNo name entered, please enter a name: ");
        gets(Lname);
    }
    for (i=0; i<count; i++)
    {
        if ( (!strcmpi(c[i].Lname,Lname)) && (!strcmpi(c[i].Fname,Fname)) )
        {
            if (j==1)
                printf("\nContacts found :\n");
            printf("\n#%d\n",j);
            printf("First name: %s\n",c[i].Fname);
            printf("Last name: %s\n",c[i].Lname);
            printf("Address: %s\n",c[i].address);
            printf("E-mail: %s\n",c[i].email);
            printf("Number: %s\n",c[i].number);
            printf("DOB: %d/%d/%d\n\n",c[i].dob.day,c[i].dob.month,c[i].dob.year);
            index[counter]=i;

```




```

        j++;
        counter++;
    }
}
if(!counter) printf("\a\nNo contacts found.\n");
else if (counter==1)
{
    answer = (char *)malloc(10);
    printf("\nDo you want to delete this contact?(y/n) >> ");
    gets(answer);
    if(CheckAnswerYes_No(answer))
    {
        for(i=index[0]; i<count; i++)
        {
            c[i]=c[i+1];
        }
        count=count-1;
        if(autosave) SaveDirectoryInFile(2);
        edit=1;
        printf("\n\aDeleted!\n");
    }
    free(answer);
}
else
{
    answer = (char *)malloc(10);
    printf("\nDo you want to delete any of these contacts?(y/n) >> ");
    gets(answer);
    if(CheckAnswerYes_No(answer))
    {
        printf("\nWhich one you want to delete? >> ");
        scanf("%d",&x);
        getchar();
        while (x>counter)
        {
            printf("\n\aWrong input, Please Enter a correct answer: ");
            scanf("%d",&x);
            getchar();
            if (x <= counter) break;
        }
        for(i=index[x-1]; i<count; i++)
        {
            c[i]=c[i+1];
        }
        count=count-1;
        if(autosave) SaveDirectoryInFile(2);
        edit=1;
        printf("\n\aDeleted!\n");
    }
    free(answer);
}
free(Lname);
free(Fname);
}

```



```

void ModifyContact() //function to modify a contact in the directory
{
    int i,x,j=1,counter=0,index[count];
    char *Lname = (char *)malloc(35),*Lname1 = (char *)malloc(35),*answer;
    printf("\nEnter last name of the contact: ");
    gets(Lname);
    while(Lname[0]!='\0')
    {
        printf("\nNo name entered, please enter a name: ");
        gets(Lname);
    }
    CorrectLettersCase(Lname);
    strcpy(Lname1,Lname);
    strlwr(Lname1);
    if (strlen(Lname) == 1)
    {
        for (i=0; i<count; i++)
        {
            for (j=0; j<(strlen(c[i].Lname)) ; j++)
            {
                if ((strstr(c[i].Lname,Lname)) || (strstr(c[i].Lname,Lname1)))
                {
                    if (j==1)
                        printf("\nContacts found :\n");
                    printf("\n#%d\n",j);
                    printf("First name: %s\n",c[i].Fname);
                    printf("Last name: %s\n",c[i].Lname);
                    printf("Address: %s\n",c[i].address);
                    printf("E-mail: %s\n",c[i].email);
                    printf("Number: %s\n",c[i].number);
                    printf("DOB: %d/%d/%d\n\n",c[i].dob.day,c[i].dob.month,c[i].dob.year);
                    index[counter]=i;
                    j++;
                    counter ++;
                    break;
                }
            }
        }
    }
    else
    {
        for(i=0; i<count; i++)
        {
            if ((strstr(c[i].Lname,Lname)) || (strstr(c[i].Lname,Lname1))))
            {
                if (j==1)
                    printf("\nContacts found :\n");
                printf("\n#%d\n",j);
                printf("First name: %s\n",c[i].Fname);
                printf("Last name: %s\n",c[i].Lname);
                printf("Address: %s\n",c[i].address);
                printf("E-mail: %s\n",c[i].email);
                printf("Number: %s\n",c[i].number);
                printf("DOB: %d/%d/%d\n\n",c[i].dob.day,c[i].dob.month,c[i].dob.year);
            }
        }
    }
}

```



```

        index[counter]=i;
        j++;
        counter ++;
    }
}
j=1;
if(!counter) printf("\a\nNo contacts found.\n");
else if (counter==1)
{
    answer = (char *)malloc(10);
    printf("Do you want to modify this contact?(y/n) >> ");
    gets(answer);
    if(CheckAnswerYes_No(answer))
    {
        if (j==1)
            printf("\nEnter modified data for the contact:\n");
        printf("First Name: ");
        gets(c[index[0]].Fname);
        CorrectLettersCase(c[index[0]].Fname);
        printf("Last Name: ");
        gets(c[index[0]].Lname);
        CorrectLettersCase(c[index[0]].Lname);
        printf("Address: ");
        gets(c[index[0]].address);
        printf("Email: ");
        gets(c[index[0]].email);
        strlwr(c[index[0]].email);
        printf("Number: ");
        gets(c[index[0]].number);
        printf("DOB (DD MM YEAR): ");
        scanf("%d %d %d",&c[index[0]].dob.day,&c[index[0]].dob.month,&c[index[0]].dob.year);
        getchar();
        printf("\nChecking for errors of the modified contact:\n");
        ErrorsOfContact();
        printf("\nNo errors or no errors remaining!\n");
        if(autosave) SaveDirectoryInFile(2);
        edit=1;
        printf("\n\aModified!\n");
    }
    free(answer);
}
else
{
    answer = (char *)malloc(10);
    printf("\nDo you want to modify any of these contacts?(y/n) >> ");
    gets(answer);
    if(CheckAnswerYes_No(answer))
    {
        printf("\nWhich one you want to modify? ");
        scanf("%d",&x);
        getchar();
        while (x>counter)
        {
            printf("\n\aWrong input, Please Enter a correct answer: ");

```



```

        scanf("%d",&x);
        getchar();
        if (x <= counter) break;
    }
    if (j==1)
        printf("\nEnter modified data for contact # %d:\n",x);
    printf("First Name: ");
    gets(c[index[x-1]].Fname);
    CorrectLettersCase(c[index[x-1]].Fname);
    printf("Last Name: ");
    gets(c[index[x-1]].Lname);
    CorrectLettersCase(c[index[x-1]].Lname);
    printf("Address: ");
    gets(c[index[x-1]].address);
    printf("Email: ");
    gets(c[index[x-1]].email);
    strlwr(c[index[x-1]].email);
    printf("Number: ");
    gets(c[index[x-1]].number);
    printf("DOB (DD MM YEAR): ");
    scanf("%d %d %d",&c[index[x-1]].dob.day,&c[index[x-1]].dob.month,&c[index[x-1]].dob.year);
    getchar();
    printf("\nChecking for errors of the modified contact:\n");
    ErrorsOfContact();
    printf("\nNo errors or no errors remaining!\n");
    if (autosave) SaveDirectoryInFile(2);
    edit=1;
    printf("\n\naModified!\n");
}
free(answer);
}
free(Lname1);
free(Lname);
}

```

void SortContactsByLname() //function to sort directory according to Lname

```

{
    contact temp;
    int i,j;
    for (i=1; i<count; i++)
    {
        for (j=0; j<count-i; j++)
        {
            if (strcmpi(c[j].Lname,c[j+1].Lname) > 0)
            {
                temp = c[j];
                c[j]=c[j + 1];
                c[j+1]=temp;
            }
        }
    }
}

```

void SortContactsByDOB() //function to sort directory according to DOB

```

{

```



```

int i,j;
contact temp;
for (i=1; i<count; i++)
{
    for (j=0; j<count-i; j++)
    {
        if (c[j].dob.year == c[j+1].dob.year)
        {
            if (c[j].dob.month == c[j+1].dob.month)
            {
                if(c[j].dob.day < c[j+1].dob.day)
                {
                    temp = c[j];
                    c[j]=c[j + 1];
                    c[j+1]=temp;
                }
            }
            else if ( c[j].dob.month < c[j+1].dob.month)
            {
                temp = c[j];
                c[j]=c[j + 1];
                c[j+1]=temp;
            }
        }
        else if (c[j].dob.year < c[j+1].dob.year )
        {
            temp = c[j];
            c[j]=c[j + 1];
            c[j+1]=temp;
        }
    }
}

void AskSorting()
{
    int x;
    printf("\nHow do you want to sort the directory?\n1.By Last name\t2.By DOB  >> ");
    scanf("%d",&x);
    getchar();
    while((x!=1) && (x!=2))
    {
        printf("\aWrong input, Please Enter a correct answer: ");
        scanf("%d",&x);
        getchar();
        if ((x==1) || (x==2)) break;
    }
    switch (x)
    {
    case 1:
        SortContactsByLname();
        break;
    case 2:
        SortContactsByLname();
        SortContactsByDOB();
    }
}

```



```

        break;
    }
}

void PrintDirectory()
{
    int i,by;
    AskSorting();
    printf("\n\nTHE PHONE DIRECTORY\n");
    printf("=====\n");
    printf("\nFirst name\tLast name\tNumber\t\tE-Mail\t\t\t\tAddress\t\t\t\t\tDOB\n");

    printf("_____
    _____\n\n");
    for(i=0; i<count; i++)
    {
        if(c[i].dob.year >= 2000) by = c[i].dob.year-2000;
        if(c[i].dob.year < 2000) by = c[i].dob.year-1900;
        printf("%-15s\t%-15s\t%-11s\t%-25s\t%-25s\t
%02d/%02d/%02d\n",c[i].Fname,c[i].Lname,c[i].number,c[i].email,c[i].address,c[i].dob.day,c[i].dob.month,by);

    printf("_____
    _____\n\n");
    }
}

void SaveDirectoryInFile(int mode)
{
    char *Filename,*answer;
    FILE *f;
    int i;
    switch (mode)
    {
    case 1:
        answer = (char *)malloc(10);
        printf("\nDo you want to save in the same file?(y/n) >> ");
        gets(answer);
        if (CheckAnswerYes_No(answer))
        {
            f=fopen(OrigFileName,"w");
            for(i=0; i<count; i++)
            {
                fprintf(f,"%s",c[i].Lname);
                fprintf(f,"%s",c[i].Fname);
                fprintf(f,"%d-%d-%d",c[i].dob.day,c[i].dob.month,c[i].dob.year);
                fprintf(f,"%s",c[i].address);
                fprintf(f,"%s",c[i].number);
                fprintf(f,"%s\n",c[i].email);
            }
            fclose(f);
            printf("\a\nSaved!");
        }
        else
        {
            Filename=(char *)malloc(25);

```



```

printf("\nEnter the new file name: ");
gets(Filename);
strcat(Filename, ".txt");
f=fopen(Filename, "w");
for(i=0; i<count; i++)
{
    fprintf(f, "%s", c[i].Lname);
    fprintf(f, "%s", c[i].Fname);
    fprintf(f, "%d-%d-%d", c[i].dob.day, c[i].dob.month, c[i].dob.year);
    fprintf(f, "%s", c[i].address);
    fprintf(f, "%s", c[i].number);
    fprintf(f, "%s\n", c[i].email);
}
fclose(f);
free(Filename);
printf("\a\nSaved!");
}
free(answer);
break;
case 2:
f=fopen(OrigFileName, "w");
for(i=0; i<count; i++)
{
    fprintf(f, "%s", c[i].Lname);
    fprintf(f, "%s", c[i].Fname);
    fprintf(f, "%d-%d-%d", c[i].dob.day, c[i].dob.month, c[i].dob.year);
    fprintf(f, "%s", c[i].address);
    fprintf(f, "%s", c[i].number);
    fprintf(f, "%s\n", c[i].email);
}
fclose(f);
break;
}
}

void QuitProgram(int mode)
{
    char *answer;
    switch(mode)
    {
        case 1:
            exit(0);
        case 2:
            answer = (char *)malloc(10);
            printf("\a\nCurrently opened file \"%s\" is not saved, Do you want to save it?(y/n) >> ", OrigFileName);
            gets(answer);
            CorrectLettersCase(answer);
            if(CheckAnswerYes_No(answer))
            {
                SaveDirectoryInFile(1);
                free(answer);
                exit(0);
            }
            else

```



```

    {
        free(answer);
        exit(0);
    }
}

void CommandsMenu()
{
    printf("Commands Menu\n");
    printf("=====\n");
    printf("Command Name\tFunction\n");
    printf("=====\t=====\n");
    printf("LOAD      \tRead data from a file\n");
    printf("QUERY     \tSearch the directory\n");
    printf("ADD       \tAdd new contacts to the directory\n");
    printf("DELETE    \tDelete a certain contact from the directory\n");
    printf("MODIFY    \tModify a certain contact into the directory\n");
    printf("PRINT     \tPrint the entire directory (sorted)\n");
    printf("SAVE      \tSave the directory into a text file (.txt)\n");
    printf("MENU      \tPrint the commands menu\n");
    printf("AUTOSAVE  \tAutosave all commands executed in the original opened file\n");
    printf("SORTF     \tSort currently opened file\n");
    printf("PRINTO    \tPrint currently opened file name\n");
    printf("LOADN     \tRead date from a new file\n");
    printf("QUIT      \tExit the program\n\n");
}

void CommandsOperators(char s[])
{
    char *answer;
    if(!strcmp(s,"quit") && !load)
        QuitProgram(1);
    if ((strcmp(s,"load") && (strcmp(s,"query") && (strcmp(s,"add") && (strcmp(s,"delete") && (strcmp(s,"modify")
&& (strcmp(s,"print") && (strcmp(s,"save") && (strcmp(s,"menu") && (strcmp(s,"autosave") && (strcmp(s,"sortf")
&& (strcmp(s,"printo") && (strcmp(s,"loadn")
&& (strcmp(s,"quit"))))
    {
        printf("\n\nWrong command or command not found!");
        return;
    }
    if (!strcmp(s,"load") && load)
    {
        printf("\n\nA file is already open.");
        return;
    }
    else if (!strcmp(s,"LOAD"))
    {
        ReadContactsFromFile();
        return;
    }
    while (load)
    {
        if ( (!strcmp(s,"autosave") && load )
        {

```




```

    autosave=1;
    printf("\n\nAutosave activated!");
    return;
}
else if ( (!strcmpi(s,"query")) && load )
{
    SearchDirectory();
    return;
}
else if ( (!strcmpi(s,"add")) && load )
{
    AddContacts();
    return;
}
else if ( (!strcmpi(s,"delete")) && load )
{
    DeleteContact();
    return;
}
else if ( (!strcmpi(s,"modify")) && load )
{
    ModifyContact();
    return;
}
else if ( (!strcmpi(s,"sortf")) && load )
{
    AskSorting();
    printf("\n\nSorted!");
    SaveDirectoryInFile(2);
    return;
}
else if ( (!strcmpi(s,"print")) && load )
{
    PrintDirectory();
    return;
}
else if ( (!strcmpi(s,"printo")) && load )
{
    printf("\n\nCurrently opened file's name is \"%s\".",OrigFileName);
    return;
}
else if ( (!strcmpi(s,"save")) && load )
{
    SaveDirectoryInFile(1);
    saved=1;
    return;
}
else if ( (!strcmpi(s,"menu")) && load )
{
    printf("\n\n");
    CommandsMenu();
    return;
}
else if ( (!strcmpi(s,"quit")) && load )
{

```



```

        if (autosave || saved || !edit) QuitProgram(1);
        else QuitProgram(2);
    }
    else if ( (!strcmpi(s,"loadn")) && load )
    {
        if (edit && !autosave && !saved)
        {
            answer = (char *)malloc(10);
            printf("\a\nCurrently opened file \"%s\" is not saved, Do you want to save it?(y/n) >> ",OrigFileName);
            gets(answer);
            CorrectLettersCase(answer);
            if(CheckAnswerYes_No(answer))
            {
                SaveDirectoryInFile(1);
                printf("\n");
                load=0;
                autosave=0;
                saved=0;
                edit=0;
                ReadContactsFromFile();
            }
            else ReadContactsFromFile();
            free(answer);
        }
        else if(!edit)
        {
            load=0;
            autosave=0;
            saved=0;
            edit=0;
            ReadContactsFromFile();
        }
        else if ((autosave || saved) && edit)
        {
            load=0;
            autosave=0;
            saved=0;
            edit=0;
            ReadContactsFromFile();
        }
        return;
    }
}
if (!load) printf("\a\nNo file opened!");
}

int main()
{
    char*answer;
    CommandsMenu();
    while(1)
    {
        answer=(char *)malloc(35);
        printf("Type a command >> ");
        gets(answer);
    }
}

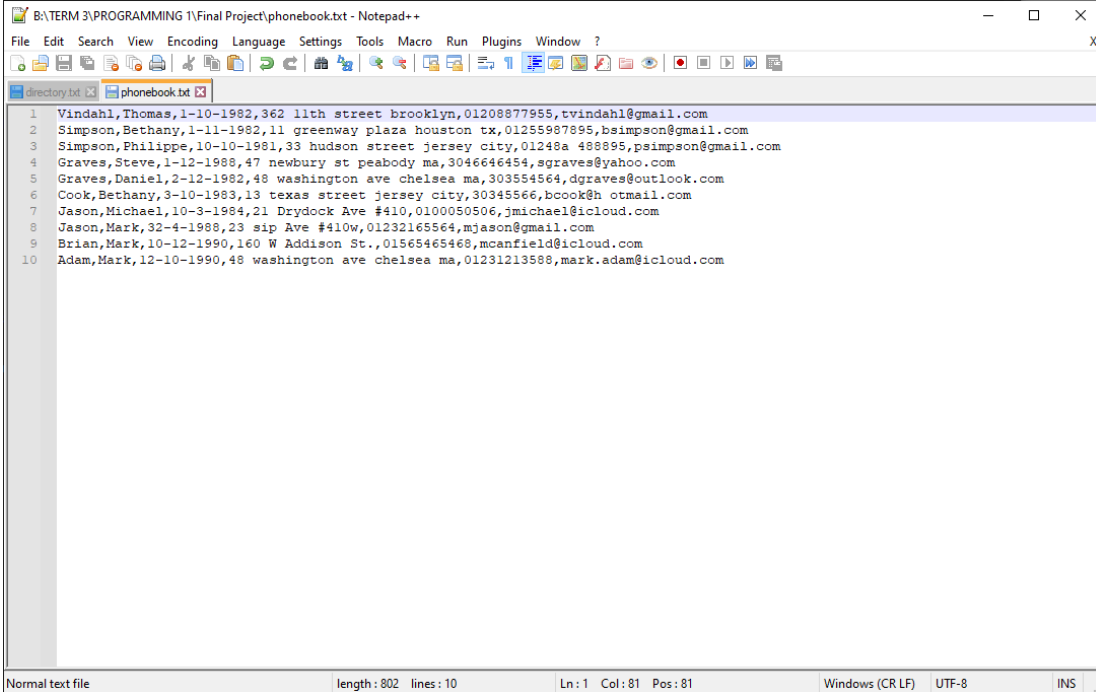
```



```
    CommandsOperators(answer);  
    free(answer);  
    printf("\n\n");  
}  
return 0;  
}
```



7 Sample runs

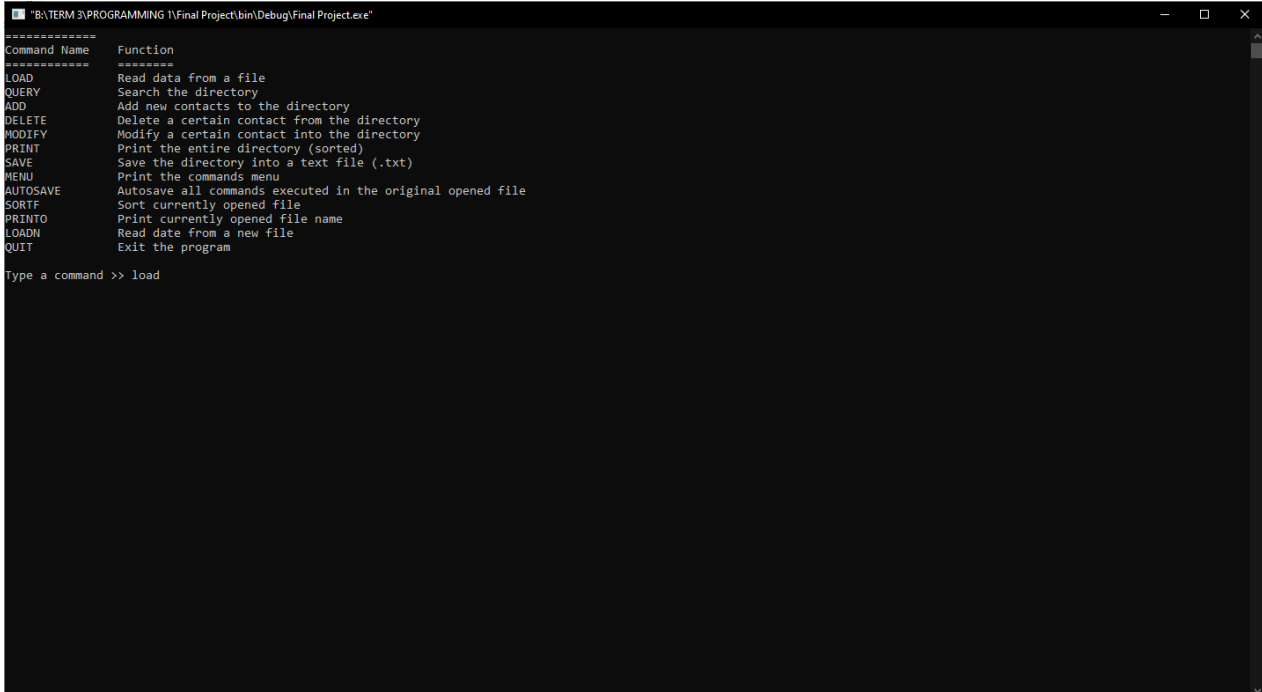


A screenshot of a Notepad++ window titled "B:\TERM 3\PROGRAMMING 1\Final Project\phonebook.txt - Notepad++". The window shows a text file named "phonebook.txt" with 10 lines of contact information. The status bar at the bottom indicates "Normal text file", "length : 802 lines : 10", "Ln : 1 Col : 81 Pos : 81", "Windows (CR LF)", "UTF-8", and "INS".

```

1 Vindahl,Thomas,1-10-1982,362 11th street brooklyn,01208877955,tvindahl@gmail.com
2 Simpson,Bethany,1-11-1982,11 greenway plaza houston tx,01255987895,bsimpson@gmail.com
3 Simpson,Philippe,10-10-1991,33 hudson street jersey city,01248a 488895,psimpson@gmail.com
4 Graves,Steve,1-12-1988,47 newbury st peabody ma,3046646454,sgraves@yahoo.com
5 Graves,Daniel,2-12-1982,48 washington ave chelsea ma,303554564,dgraves@outlook.com
6 Cook,Bethany,3-10-1983,13 texas street jersey city,30345566,bcook@h otmail.com
7 Jason,Michael,10-3-1984,21 Drydock Ave #410,0100050506,jmichael@icloud.com
8 Jason,Mark,32-4-1988,23 sip Ave #410w,01232165564,mjason@gmail.com
9 Brian,Mark,10-12-1990,160 W Addison St.,01565465468,mcanfield@icloud.com
10 Adam,Mark,12-10-1990,48 washington ave chelsea ma,01231213588,mark.adam@icloud.com

```



A screenshot of a command-line interface window titled "B:\TERM 3\PROGRAMMING 1\Final Project\bin\Debug\Final Project.exe". The window displays a list of commands and their functions. The prompt "Type a command >> load" is visible at the bottom.

```

=====
Command Name      Function
=====
LOAD              Read data from a file
QUERY             Search the directory
ADD               Add new contacts to the directory
DELETE            Delete a certain contact from the directory
MODIFY            Modify a certain contact into the directory
PRINT             Print the entire directory (sorted)
SAVE              Save the directory into a text file (.txt)
MENU              Print the commands menu
AUTOSAVE          Autosave all commands executed in the original opened file
SORTF             Sort currently opened file
PRINTO            Print currently opened file name
LOADN             Read data from a new file
QUIT             Exit the program

Type a command >> load

```



```

"\"B:\\TERM 3\\PROGRAMMING 1\\Final Project\\bin\\Debug\\Final Project.exe"
=====
Command Name      Function
=====
LOAD              Read data from a file
QUERY            Search the directory
ADD              Add new contacts to the directory
DELETE           Delete a certain contact from the directory
MODIFY           Modify a certain contact into the directory
PRINT            Print the entire directory (sorted)
SAVE             Save the directory into a text file (.txt)
MENU            Print the commands menu
AUTOSAVE         Autosave all commands executed in the original opened file
SORTF           Sort currently opened file
PRINTO          Print currently opened file name
LOADN           Read data from a new file
QUIT            Exit the program

Type a command >> load

Enter file path (txt files only): phonebook.txt

Checking for errors contact by contact:

Contact Mark Jason with birth day "32", has wrong birth day, birth day must be between 1 and 31.
Enter the correct birth day: 31

Contact Philippe Simpson with phone number "01248a 488895", has wrong number, phone number must contain only numbers.
Enter the correct number: 01248488895

Contact Bethany Cook with e-mail "bcook@h otmail.com", has wrong e-mail, e-mail must not contain spaces.
Enter the correct e-mail: bcook@hotmail.com

No errors or no errors remaining!

File read successfully!

Type a command >>

```

```

"\"B:\\TERM 3\\PROGRAMMING 1\\Final Project\\bin\\Debug\\Final Project.exe"
Type a command >> load

Enter file path (txt files only): phonebook.txt

Checking for errors contact by contact:

Contact Mark Jason with birth day "32", has wrong birth day, birth day must be between 1 and 31.
Enter the correct birth day: 31

Contact Philippe Simpson with phone number "01248a 488895", has wrong number, phone number must contain only numbers.
Enter the correct number: 01248488895

Contact Bethany Cook with e-mail "bcook@h otmail.com", has wrong e-mail, e-mail must not contain spaces.
Enter the correct e-mail: bcook@hotmail.com

No errors or no errors remaining!

File read successfully!

Type a command >> query

Enter last name of the contact: simpson

Contacts found :

#1
First name: Bethany
Last name: Simpson
Address: 11 greenway plaza houston tx
E-mail: bsimpson@gmail.com
Number: 01255987895
DOB: 1/11/1982

#2
First name: Philippe
Last name: Simpson
Address: 33 hudson street jersey city
E-mail: psimpson@gmail.com
Number: 01248488895
DOB: 10/10/1981

Type a command >>

```



```

B:\TERM 3\PROGRAMMING 1\Final Project\bin\Debug\Final Project.exe
Enter last name of the contact: simpson
Contacts found :
#1
First name: Bethany
Last name: Simpson
Address: 11 greenway plaza houston tx
E-mail: bsimpson@gmail.com
Number: 01255987895
DOB: 1/11/1982
#2
First name: Philippe
Last name: Simpson
Address: 33 hudson street jersey city
E-mail: psimpson@gmail.com
Number: 01248488895
DOB: 10/10/1981

Type a command >> delete
Enter first name of the contact: philippe
Enter last name of the contact: simpson
Contacts found :
#1
First name: Philippe
Last name: Simpson
Address: 33 hudson street jersey city
E-mail: psimpson@gmail.com
Number: 01248488895
DOB: 10/10/1981

Do you want to delete this contact?(y/n) >> y
Deleted!

Type a command >> _

```

```

B:\TERM 3\PROGRAMMING 1\Final Project\bin\Debug\Final Project.exe
Do you want to delete this contact?(y/n) >> y
Deleted!

Type a command >> print
How do you want to sort the directory?
1.By Last name 2.By DOB >> 2

THE PHONE DIRECTORY
*****

```

First name	Last name	Number	E-Mail	Address	DOB
Mark	Brian	01565465468	mcanfield@icloud.com	160 W Addison St.	10/12/90
Mark	Adam	01231213588	mark.adam@icloud.com	48 washington ave chelsea ma	12/10/90
Steve	Graves	3046646454	sgraves@yahoo.com	47 newbury st peabody ma	01/12/88
Mark	Jason	01232165564	mjason@gmail.com	23 sip Ave #410w	31/04/88
Michael	Jason	0100050506	j michael@icloud.com	21 Drydock Ave #410	10/03/84
Bethany	Cook	30345566	bcCook@hotmail.com	13 texas street jersey city	03/10/83
Daniel	Graves	303554564	dgraves@outlook.com	48 washington ave chelsea ma	02/12/82
Bethany	Simpson	01255987895	bsimpson@gmail.com	11 greenway plaza houston tx	01/11/82
Thomas	Vindahl	01208877955	tvindahl@gmail.com	362 11th street brooklyn	01/10/82

```

Type a command >> _

```



```

"R:\TERM 3\PROGRAMMING 1\Final Project\bin\Debug\Final Project.exe"
1.By Last name 2.By DOB  >> 2

THE PHONE DIRECTORY
=====
First name    Last name    Number      E-Mail      Address      DOB
-----
Mark          Brian        01565465468 mcanfield@icloud.com 160 W Addison St. 10/12/90
Mark          Adam         01231213588 mark.adam@icloud.com 48 washington ave chelsea ma 12/10/90
Steve         Graves       3046646454  sgraves@yahoo.com 47 newbury st peabody ma 01/12/88
Mark          Jason        01232165564  mjason@gmail.com 23 sip Ave #410w 31/04/88
Michael       Jason        0100950506  jmichael@icloud.com 21 Drydock Ave #410 10/03/84
Bethany       Cook         30345566   bcook@hotmail.com 13 texas street jersey city 03/10/83
Daniel        Graves       303554564  dgraves@outlook.com 48 washington ave chelsea ma 02/12/82
Bethany       Simpson      01255987895 bsimpson@gmail.com 11 greenway plaza houston tx 01/11/82
Thomas        Vindahl      01288877955 tvindahl@gmail.com 362 11th street brooklyn 01/10/82

Type a command >> save
Do you want to save in the same file?(y/n) >> n
Enter the new file name: phonebook modified
Saved!
Type a command >> _

```

```

"R:\TERM 3\PROGRAMMING 1\Final Project\bin\Debug\Final Project.exe"
=====
First name    Last name    Number      E-Mail      Address      DOB
-----
Mark          Brian        01565465468 mcanfield@icloud.com 160 W Addison St. 10/12/90
Mark          Adam         01231213588 mark.adam@icloud.com 48 washington ave chelsea ma 12/10/90
Steve         Graves       3046646454  sgraves@yahoo.com 47 newbury st peabody ma 01/12/88
Mark          Jason        01232165564  mjason@gmail.com 23 sip Ave #410w 31/04/88
Michael       Jason        0100950506  jmichael@icloud.com 21 Drydock Ave #410 10/03/84
Bethany       Cook         30345566   bcook@hotmail.com 13 texas street jersey city 03/10/83
Daniel        Graves       303554564  dgraves@outlook.com 48 washington ave chelsea ma 02/12/82
Bethany       Simpson      01255987895 bsimpson@gmail.com 11 greenway plaza houston tx 01/11/82
Thomas        Vindahl      01288877955 tvindahl@gmail.com 362 11th street brooklyn 01/10/82

Type a command >> save
Do you want to save in the same file?(y/n) >> n
Enter the new file name: phonebook modified
Saved!
Type a command >> quit
Process returned 0 (0x0)   execution time : 131.751 s
Press any key to continue.

```



8 User Manual

8.1 Installed commands

Command name	Function
LOAD	Read data from a file (.txt files only)
QUERY	Search for contacts in the directory
ADD	Add “ <i>n</i> ” contacts to the directory
DELETE	Delete certain contact from the directory
MODIFY	Modify certain contact in the directory
PRINT	Print the entire directory in table form
SAVE	Save the directory in a file (.txt files only)
MENU	Print the entire commands menu
AUTOSAVE	Activate autosave mode
SORTF	Sort data in the file itself
PRINTO	Print currently opened file
LOADN	Read data from a new file (.txt files only)
QUIT	Exits the program

- User can type the command in any case whether upper or lower, for example “load”, “LOAD”, “lOaD”, all of them will work.



8.2 First step

In this program the first thing you will see is the commands menu, like in the following picture:

```

B:\TERM 3\PROGRAMMING 1\Final Project\bin\Debug\Final Project.exe
Commands Menu
=====
Command Name      Function
=====
LOAD              Read data from a file
QUERY             Search the directory
ADD               Add new contacts to the directory
DELETE            Delete a certain contact from the directory
MODIFY            Modify a certain contact into the directory
PRINT             Print the entire directory (sorted)
SAVE              Save the directory into a text file (.txt)
MENU              Print the commands menu
AUTOSAVE          Autosave all commands executed in the original opened file
SORTF             Sort currently opened file
PRINTO            Print currently opened file name
LOADN             Read date from a new file
QUIT             Exit the program

Type a command >> _

```

- User must first apply “**LOAD**” command to read data from file, program will not execute any other command before completing the “**LOAD**” command. If user tries to type any other command before “**LOAD**” will encounter an error “**No file opened!**”.
- After completing the “**LOAD**” command, user can freely choose any other command from program installed commands.

“LOAD’s operation way will be known in the section (8.3)”



8.3 Commands' operation way

1- LOAD:

- User types **“LOAD”** in commands area.
- User then is asked to enter a text (.txt) file path, for example **“C:\Users\amrya\Downloads\phonebook.txt”**.
- If the file in the same program's folder, user can type only the file name, for example **“phonebook.txt”**.

“It is preferred to put the text file in same program's folder.”

- Program automatically will check for errors contact by contact (fields of errors are the phone number, e-mail, and dob. Explained in detail in section (8.4)). If any errors found user will be asked to enter the correct data till no errors found in all contacts.

2- QUERY:

- User types **“QUERY”** in commands area.
- User then is asked to enter a name (which is last name).
- Program will search for contacts with similar last name and prints data of each contact found.
- If no contacts found, program prints **“No contacts found!”**.



3- ADD:

- User types “**ADD**” in commands area.
- User then is asked to enter number of desired contacts to add.
- User will be asked to fill data of new contact field by field (last name, first name, DOB, address, e-mail, and phone number).
- Program automatically will check for errors contact by contact (fields of errors are the phone number, e-mail, and dob. Explained in detail in section (8.4)). If any errors found user will be asked to enter the correct data till no errors found in all contacts.

4- DELETE:

- User types “**DELETE**” in commands area.
- User then is asked to enter a name (last name) and another name (first name).
- Program will search for contacts with similar full name and prints data of each contact found.
- User will be asked whether to delete found contacts or not (y/n questions).
- If more than one contact found, user will be asked which one to delete (user answers by shown number of contacts).
- Afterwards, program will remove data of chosen contact completely from the directory.



5- Modify:

- User types “**Modify**” in commands area.
- User then is asked to enter a name (which is last name).
- Program will search for contacts with similar last name and prints data of each contact found.
- User will be asked whether to modify found contacts or not (y/n question).
- If more than one contact found, user will be asked which one to modify (user answers by shown number of contacts).
- Afterwards, user will be asked to fill modified data of chosen contact field by field (last name, first name, DOB, address, e-mail, and phone number).
- Program automatically will check for errors contact by contact (fields of errors are the phone number, e-mail, and dob. Explained in detail in section (8.4)). If any errors found user will be asked to enter the correct data till no errors found in all contacts.

6- PRINT:

- User types “**PRINT**” in commands area.
- User then is asked in which way of sorting (by last name – by dob) the directory will be sorted.
- Program will print the entire directory sorted according to user’s choice.



7- SAVE:

- User types **“SAVE”** in commands area.
- User then is asked whether to save in same file or not (y/n question).
- If user chooses to save in same file, program automatically saves in the opened file.
- If user chooses to save in new file, then user is asked to enter a text (.txt) file name, user should only type the file name (filename not filename.txt) and file will be saved in same program's folder.

8- MENU:

- User types **“MENU”** in commands area.
- Programs will provide user with complete commands menu.

9- AUTOSAVE:

- User types **“AUTOSAVE”** in commands area.
- Program will activate autosave mode which will save each modification done in the program automatically.



10- SORTF:

- User types “**SORTF**” in commands area.
- User then is asked in which way of sorting (by last name – by dob) the directory will be sorted.
- Program will sort data in the opened file itself according to user choice.

11- PRINTO:

- User types “**PRINTO**” in commands area.
- Program will print currently opened file name.

12- LOADN:

- User types “**LOADN**” in commands area.
- If the currently opened file is modified and not saved user will be asked whether to save the modifications or not (y/n question).
- User then is asked to enter a text (.txt) file path, for example “**C:\Users\amrya\Downloads\phonebook.txt**”.
- If the file in the same program’s folder, user can type only the file name, for example “**phonebook.txt**”.
- Program automatically will check for errors contact by contact (fields of errors are the phone number, e-mail, and dob. Explained



in detail in section (8.4)). If any errors found user will be asked to enter the correct data till no errors found in all contacts.

13- QUIT:

- User types “**QUIT**” in commands area.
- If the currently opened file is modified and not saved user will be asked whether to save the modifications or not (y/n question).
- Program ends.

Important notes:

- In (y/n questions) user can answer by (yes – y - no – n).
- In sorting questions user can answer by (1 or 2).
- E-mails should follow (example@domain.top-level-domain).
- Available top-level domains are (.com - .net - .org).



8.4 Field of errors

1- Phone number:

- Phone number contains characters other than numbers (0-9).

2- E-mail:

- E-mail does not contain “@” or contains more than one “@”.
- E-mail contains spaces.
- E-mail’s prefix is more than 64 characters.
- E-mail’s prefix first character is not an ascii letter (a-z) or a number (0-9).
- E-mail’s prefix contains consecutive periods “.”, dashes “-”, or underscores “_”.
- E-mail’s prefix contains characters other than ascii letters (a-z), numbers (0-9), periods (.), dashes (-) and underscores (_).
- E-mail’s prefix last character is not as ascii letter (a-z) or a number (0-9).
- E-mail’s domain is more than 253 characters.
- E-mail’s domain first character is not an ascii letter (a-z) or a number (0-9).
- E-mail’s domain does not contain any periods (.).
- E-mail’s domain contains consecutive periods “.” or dashes “-”.
- E-mail’s domain contains characters other than ascii letters (a-z), numbers (0-9), periods (.), and dashes (-).
- E-mail’s top-level domain is not one of (.com, .net, or .org).

3- Date of birth:

- Birth day is less than 1 or more than 31.
- Birth month is less than 1 or more than 12.
- Birth year is less than 1903 or more than 2021.

“Oldest person alive is the Japanese Kana Tanaka born on 2 January 1903”



9 References

[https://en.wikipedia.org/wiki/C_\(programming_language\)#Relations to other languages](https://en.wikipedia.org/wiki/C_(programming_language)#Relations_to_other_languages)

<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwilqdCPvrjuAhUO2BoKHxs4DeoQFjADegQICRA&url=https%3A%2F%2Fwww.guinnessworldrecords.com%2Fnews%2F2020%2F10%2Fthe-worlds-oldest-people-and-their-secrets-to-a-long-life-632895&usg=AOvVaw1DCn-sx6b90nhlAn2HwRpC>

<https://www.slideshare.net/thanksforvisitinghere/telephone-directory-in-c>

<https://www.geeksforgeeks.org/program-calculate-age/>

https://help.xmatters.com/ondemand/trial/valid_email_format.htm

<https://help.returnpath.com/hc/en-us/articles/220560587-What-are-the-rules-for-email-address-syntax->

<https://app.diagrams.net/>

