DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

KATHMANDU UNIVERSITY

COMP 102: Project on Telephone Directory

Submitted by:

Shishir Kafle, Harichandra Khatri, Saloon Gautam, Sanskar Karki

Department of Computer Science

School of Science, Kathmandu University

Submitted to:

Manish Joshi / Pratit Raj Giri

Lecturer

Department of Computer Science and Engineering

School of Engineering, Kathmandu University

Date:17/08/2021

Table of Content

Introduction

C is a procedural programming language. It was initially developed by Dennis Ritchie in the 1972. It was mainly developed as a system programming language to write an operating system. The main features of the C language include low-level memory access, a simple set a keyword, and a clean style, these features make C language suitable for system programming like an operating system or compiler development.

We've covered several aspects of the C programming language so far, and we're ready to develop and run simple programs. However, before attempting to create sophisticated programs, it is worthwhile to explore some programming principles that will assist in the creation of efficient and error-free programs.

Program design, program coding, and program testing are all significant steps in the development process. All three stages are necessary for a high-quality program to be produced.

We did system design, source coding, program testing, and many other aspects in “TELEPHONE DIRECTORY” to provide the greatest user experience possible. We've offered the user the ability to enter the person's information, as well as the ability to save, sort, append, delete and more… that information, among other things.

We can improve the efficiency of the system and thereby overcome the shortcomings of the current system. Some of the benefits of utilizing C in our project include:

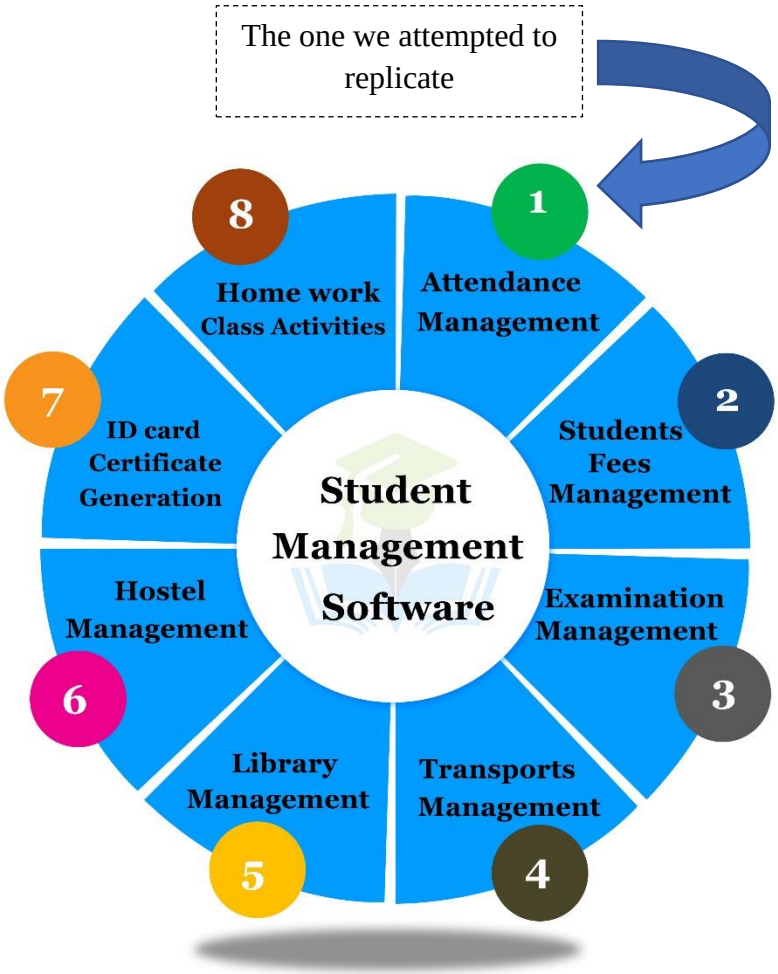
- Less human mistake is possible
- Manual labor strength and strain can be decreased
- High security;
- Data consistency
- Easy handling;
- Easy data updating;
- Easy record keeping;
- Backup data can be quickly generated.

Case Study of Similar Application

1. Student Management System.

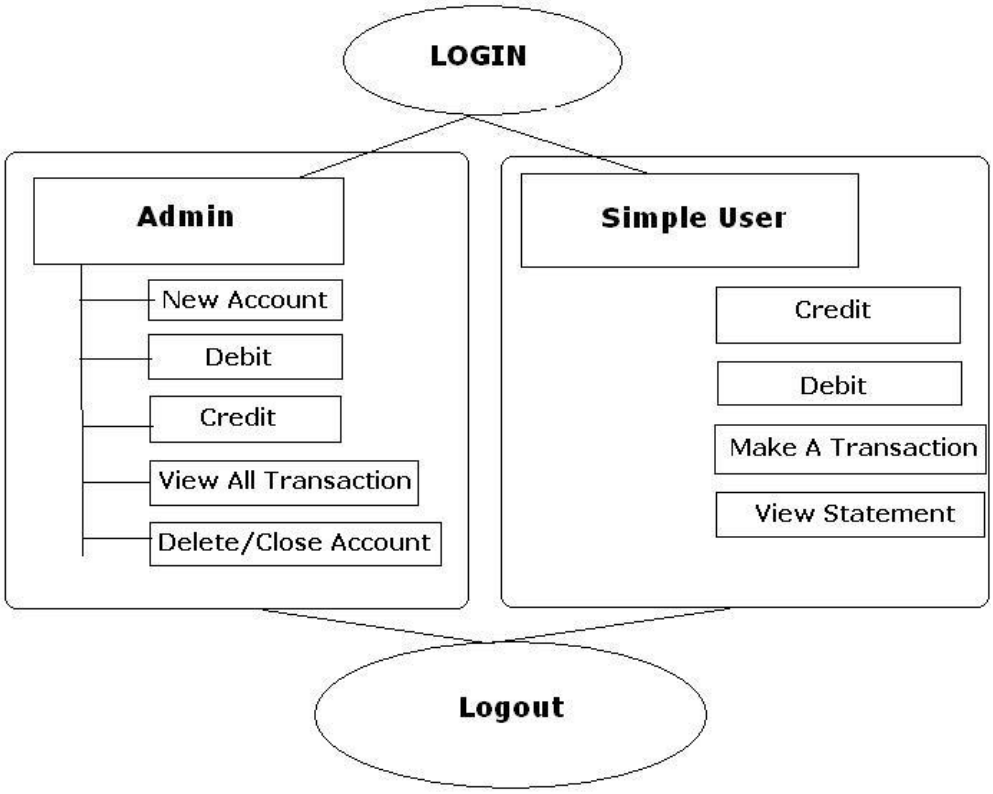
We discovered the similar technique in the student management system after being asked to sort the names of the telephone entries. The students' names, for example, are sorted by first name, last name, or roll number. In a similar way, our program allows the user to do the same thing. The user has the option of sorting the list by first or last name. Additionally, the user has the option of sorting the numbers in ascending order.

We're all aware that sorting makes finding any form of data a breeze. This may be handy if a user wants to find out how many people have the initial name "A" because he can access them directly.



2. Bank Management System

One of the most important features that every user desire is the security of their work. It might be found in their financial system or in their phone book. None of the users wants their saved phone numbers to be stolen or shared with others. As a result, we sought to incorporate the login notion into our software "TELEPHONE DIRECTORY" after extracting an idea from the bank administration system. Users must first register in the database, which is a text file in this case, and then log in to gain access to the whole telephone directory. In comparison to our telephone directory, the bank management system gives higher security. Furthermore, our program lacks double authentication, although the bank management system does. As a result, we will be able to develop this feature into our program in the future.



3. Customer Data Management System

Take, for example, any Nepalese insurance company. One of that company's customers has a ten-year accident life insurance policy, and the service given to that customer will be unavailable. As a result, the company's admin interface may not require that Customer's details so, the organization now deletes obsolete data from its database. In a similar way, our “TELEPHONE DIRECTORY” allows the registered user (in this example, admin) to delete data that is no longer needed. We can also append fresh data to our database, much like an insurance company does with newly enrolled customers' information.

Let's look at another scenario,

One of the pre-registered customers visits the Insurance Company for another policy, and when the name is submitted, the outcome displays pre-registered user. As a result, in this instance, the Insurance Company adds the customer's desired insurance to the same previous information. Our application "TELEPHONE DIRECTORY" performs something similar. It will tell you whether or not the number exists. However, when looking at the case of an insurance company adding another policy to the same individual, we notice that we are missing the ability to add multiple contact numbers for the same person, which will be addressed in the near future.

Observing the idea of a Database and repurposing it

Library used

While coding the "TELEPHONE DIRECTORY," we employed a number of header files and Datatypes, such as:

- #include <stdio.h>

{

<Stdio.h> is a header file in C, it is the file which contains C declaration and Macro definition to be shared between several files. stdio.h means standard input/output function which contains:

- printf ()
- Scanf ()
- getc ()
- putc ()
- fopen ()
- fclose ()
- fprintf ()
- fscanf ()

}

- #include<stdlib.h>

{

<stdlib.h> header file stands for Standard Library. It has the information of memory allocation/freeing functions Some predefined function that we used are:

- malloc ()
- exit ()

}

- #include<string.h>

{

<string.h> is the header in the C standard library which contains macro definitions, constants and declarations of functions and types used not only for string handling but also various memory handling functions, some function used under the <string.h> header file are:

- strcat ()
- strcmp ()
- strcpy ()
- strlen ()

}

- #include<ctype.h>

{

<ctype.h> header file of the C Standard Library declares several functions that are useful for testing and mapping characters. All the functions accept int as a parameter, whose value must be EOF or representable as an unsigned char. All the functions return non-zero (true) if the argument c satisfies the condition described, and zero(false) if not. Some pre-defined functions used under <ctype.h> header file is:

- isalpha ()
- tolower ()
- toupper ()

}

- #include<windows.h>

{

<windows.h>is a Windows-specific header file for the C programming languages which contains declarations for all of the functions in the Windows API, all the common macros used by Windows programmers, and all the data types used by the various functions and subsystems.

- Sleep ()

}

Flowchart and Algorithm

1. Algorithm:

Step 1: Start.

Step 2: Display the login options on the screen.

Step 3: Read username, password and save to .txt file.

Step 4: If login info matches then

- Print “Hello, username Welcome to the Program”

Step 5: Prints “Welcome to our telephone directory”

- Print phone records
- Add records
- Retrieve records
- Delete records
- Load file
- Save to file
- Sort
- Exit

Step 6: Now for printRec ()

- Prints First Name, Last Name, Phone Number from the Records.txt file.

Step 7: Now for addRecord ()

- Reads First Name, Last Name, Phone Number and saves to Record.txt file

Step 8: Now for rectrieveRec(),

- Reads First Name or Last Name or Phone number to search and stores in *attr
- If(Strcmp(fname, attr)==0 || strcmp(lname, attr)==0 ||strcmp(pnum, attr)==0)

- Prints “First Name, Last Name, Phone Number”

Step 9: Now for deleteRec(),

- Reads First Name or Last Name or Phone number to delete and stores in *attr
- If(Strcmp(fname, attr)==0 || strcmp(lname, attr)==0 ||strcmp(pnum, attr)==0)
- Deletes the record.

Step 10: for loadRec()

- Loads the files data to the console.

Step 11: Now for saveRec (), saves the entered data to file,

- File *wFile = fopen(“Records.txt”, “w”);
- Fprintf (“First Name, Last name, Phone number”);

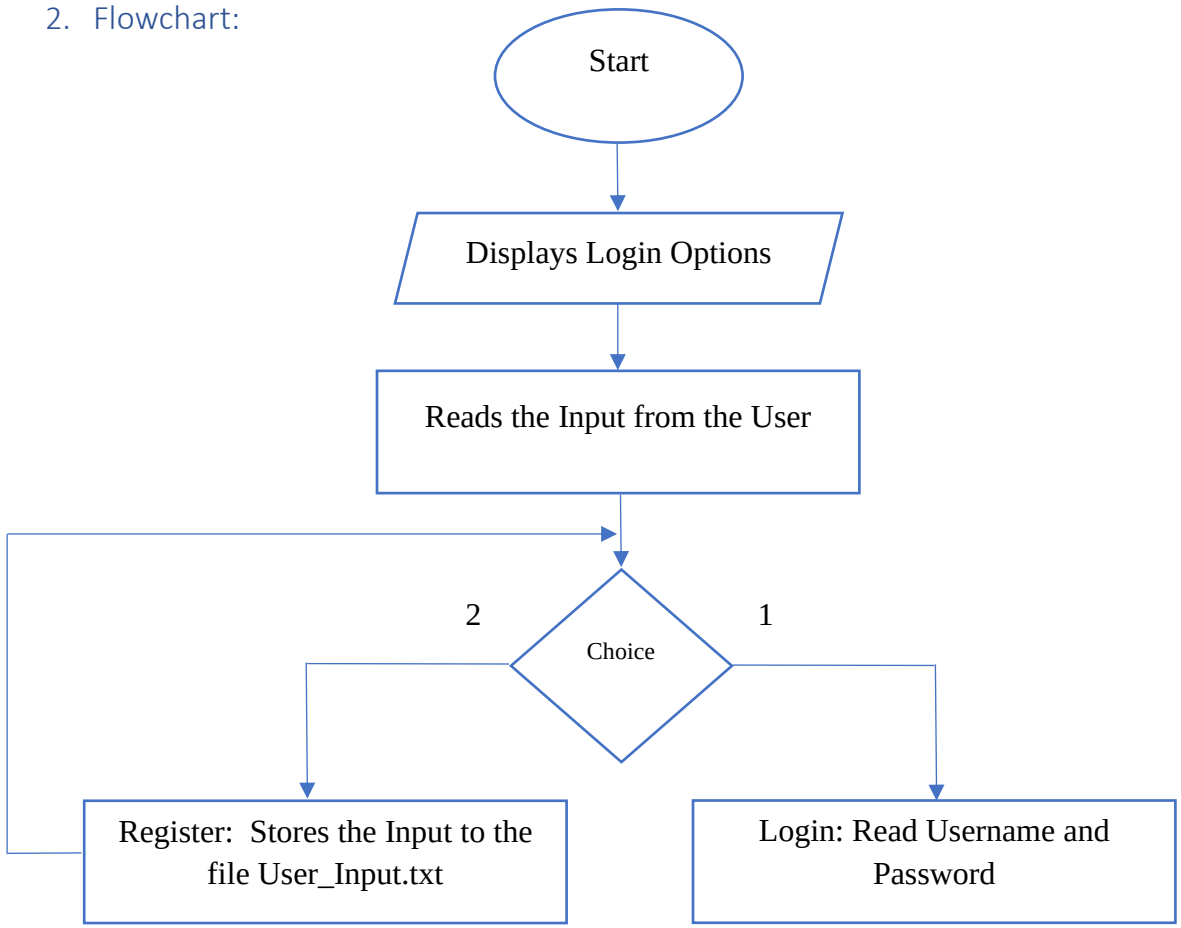
Step 12: Now for sortRec()

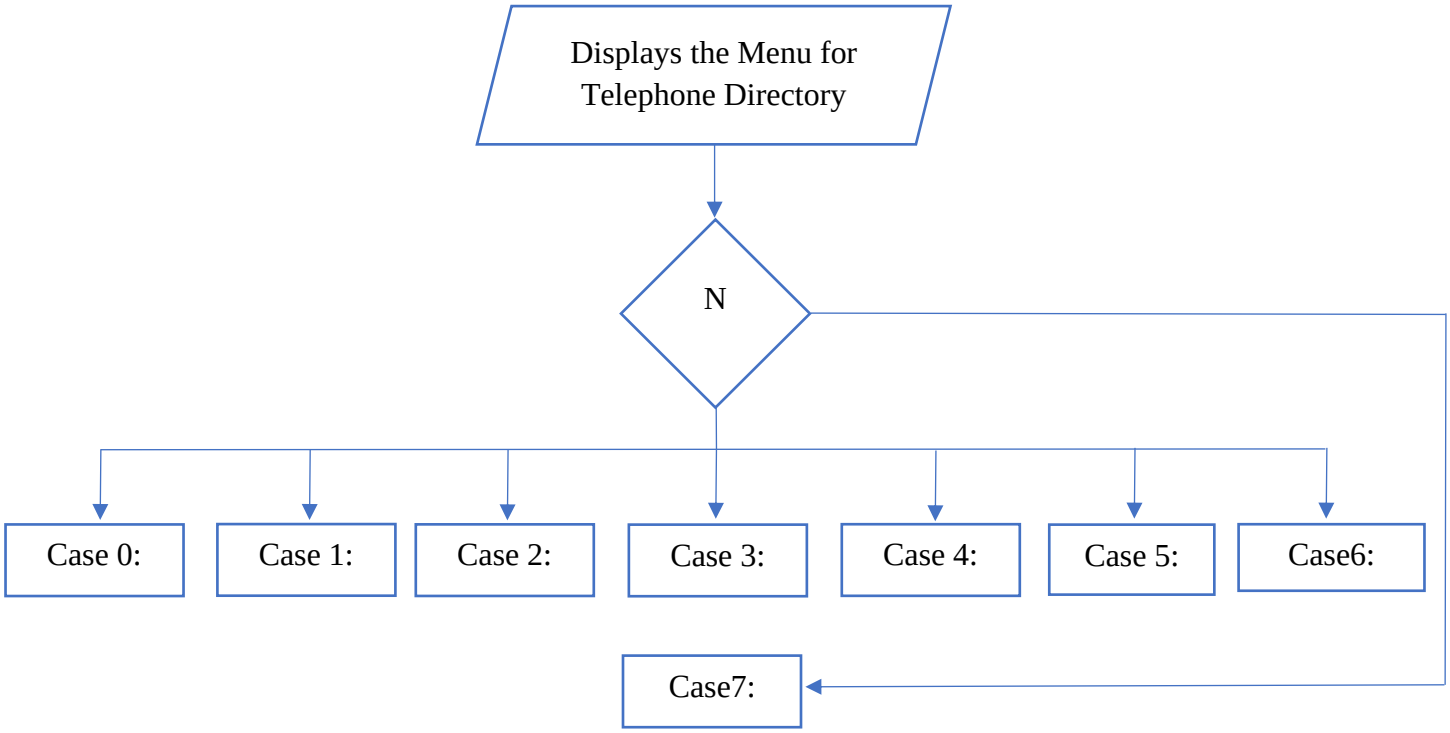
- If ‘f’ sorts by first name
- If ‘l’, sorts by last name
- If ‘p’ sorts by phone number

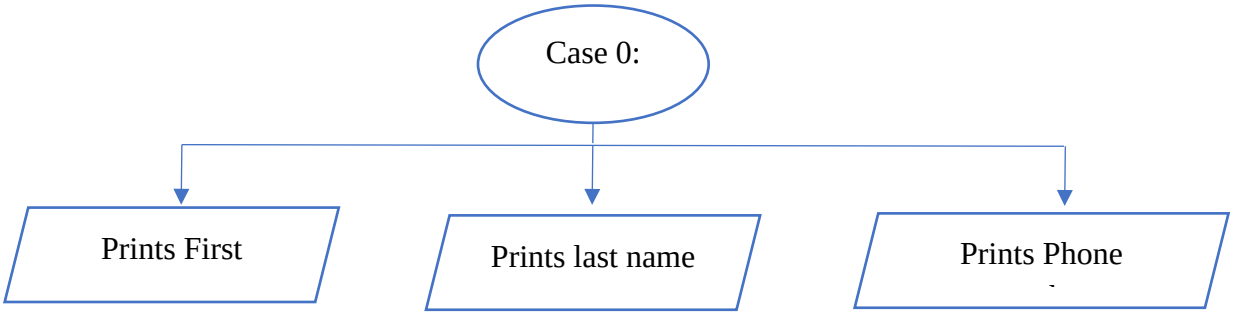
Step 13: For Exit(),

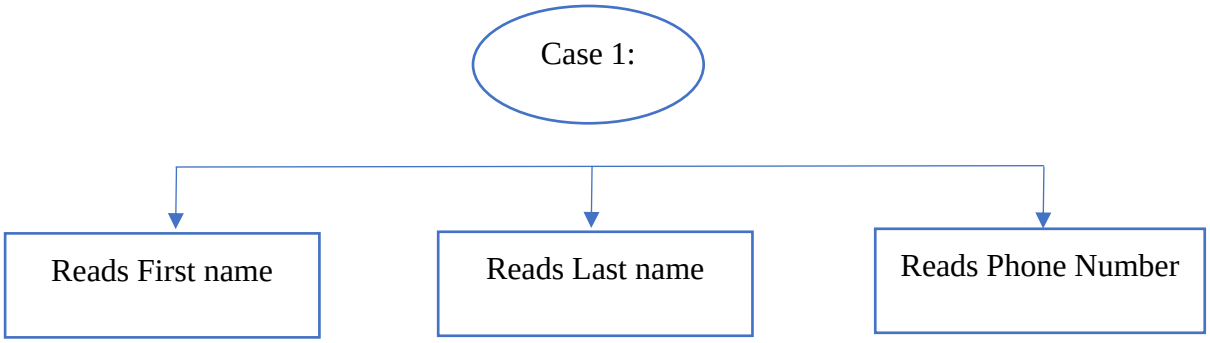
- Prints “Do you want to save data?”
- If ‘y’, saves and exit.
- If ‘n’, exits without saving changes.

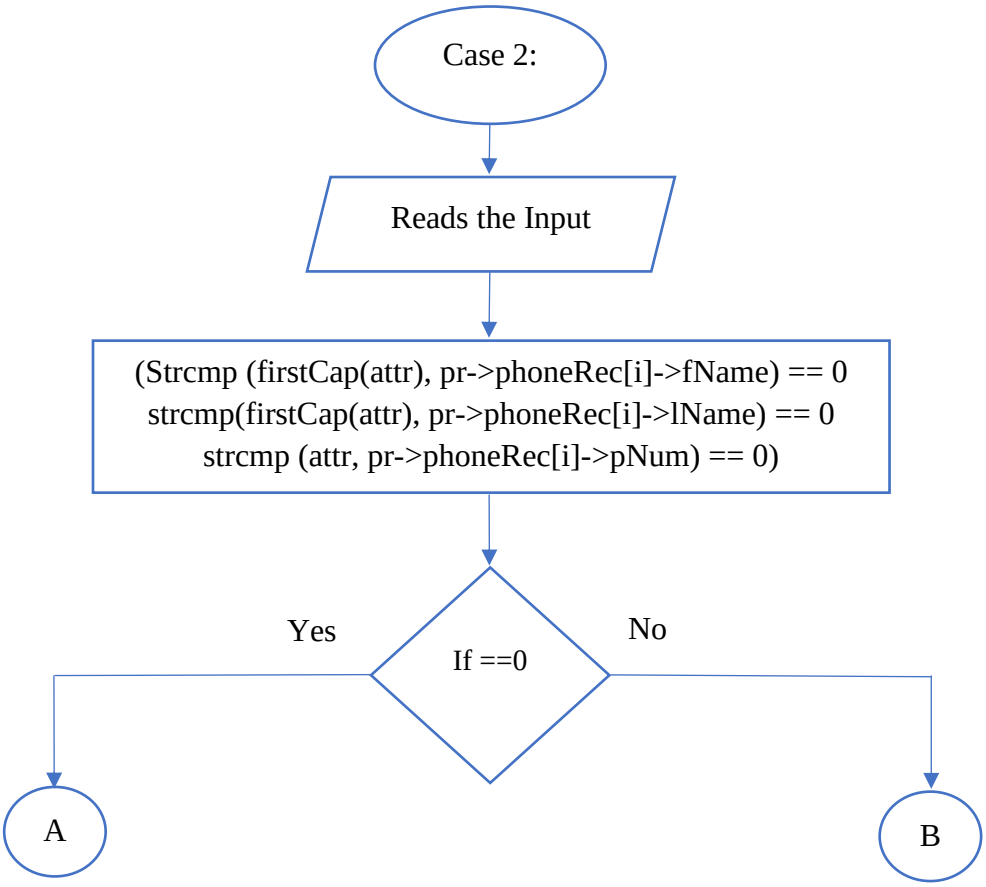
2. Flowchart:

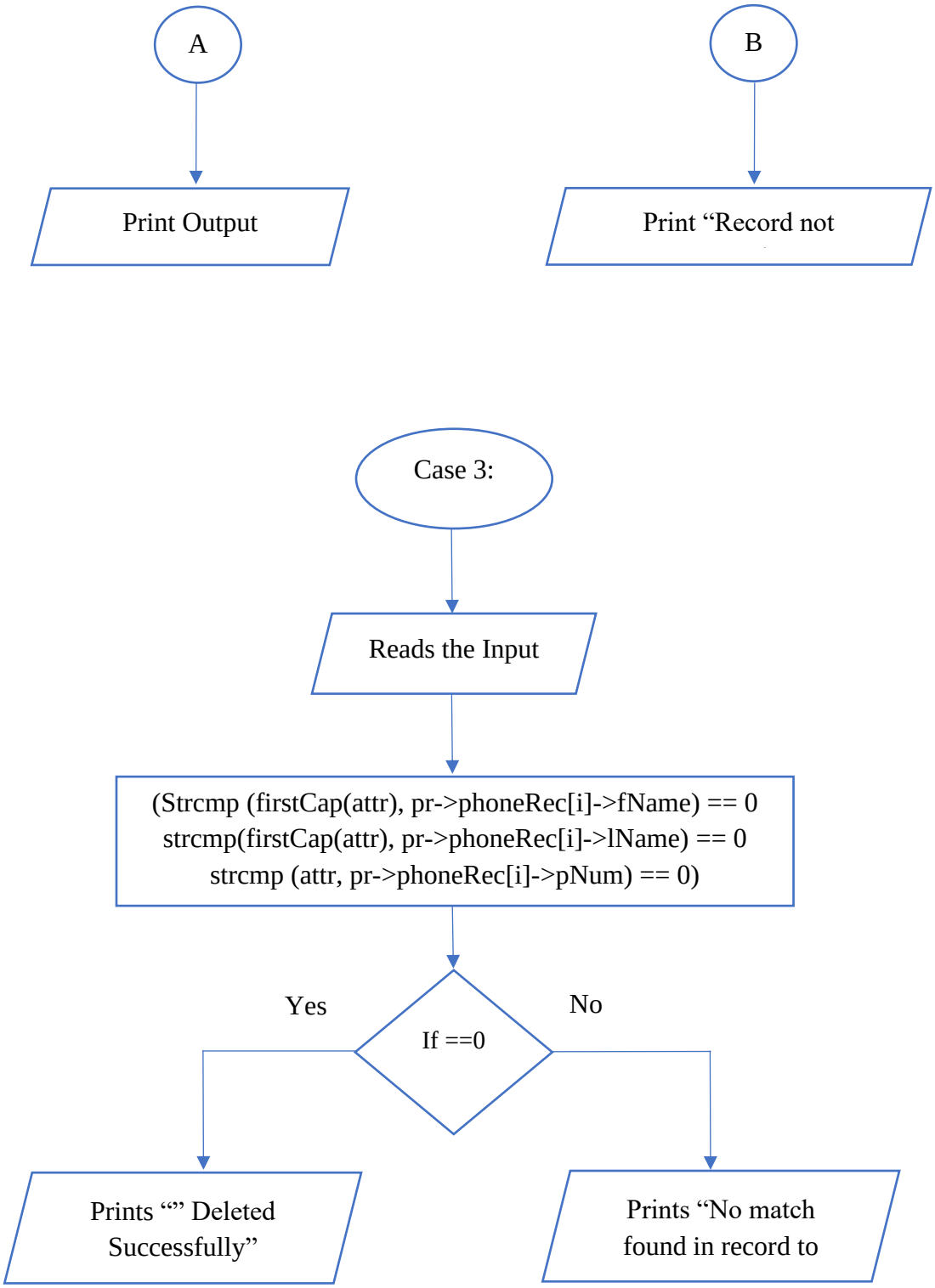

Start
Displays Login Options
Reads the Input from the User
2
1
Choice
Register: Stores the Input to the file User_Input.txt
Login: Read Username and Password

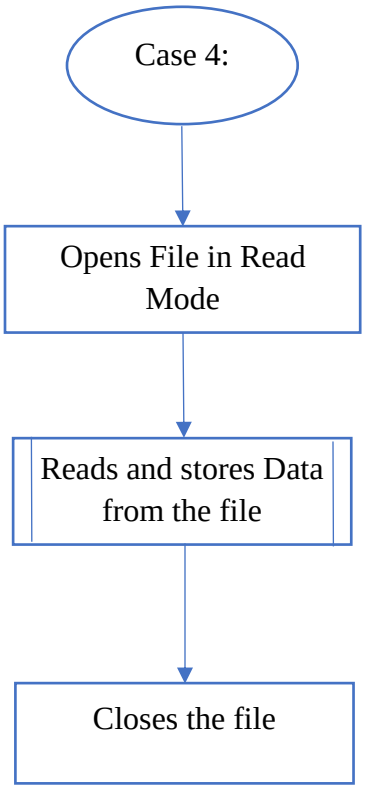

Displays the Menu for Telephone Directory
N
Case 0:
Case 1:
Case 2:
Case 3:
Case 4:
Case 5:
Case6:
Case7:

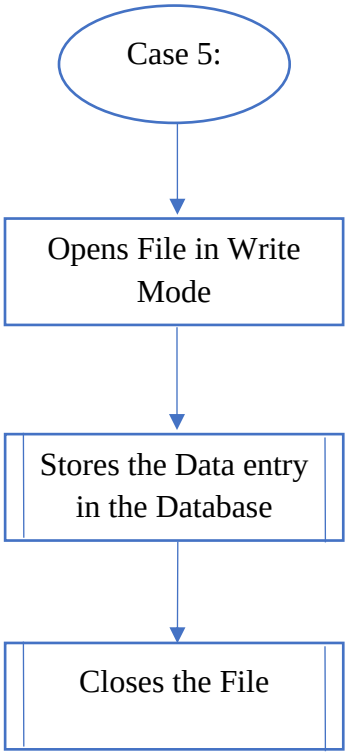

Case 0:
Prints First
Prints last name
Prints Phone

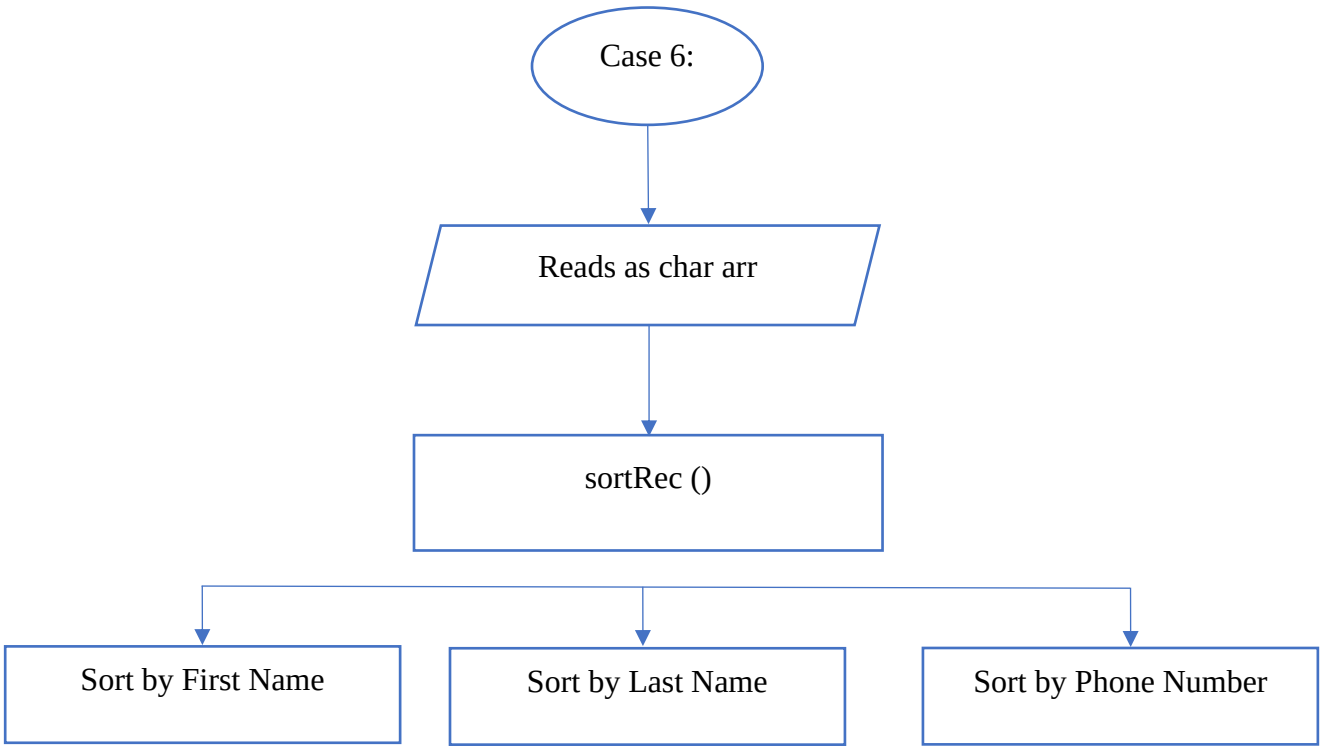

Case 1:
Reads First name
Reads Last name
Reads Phone Number

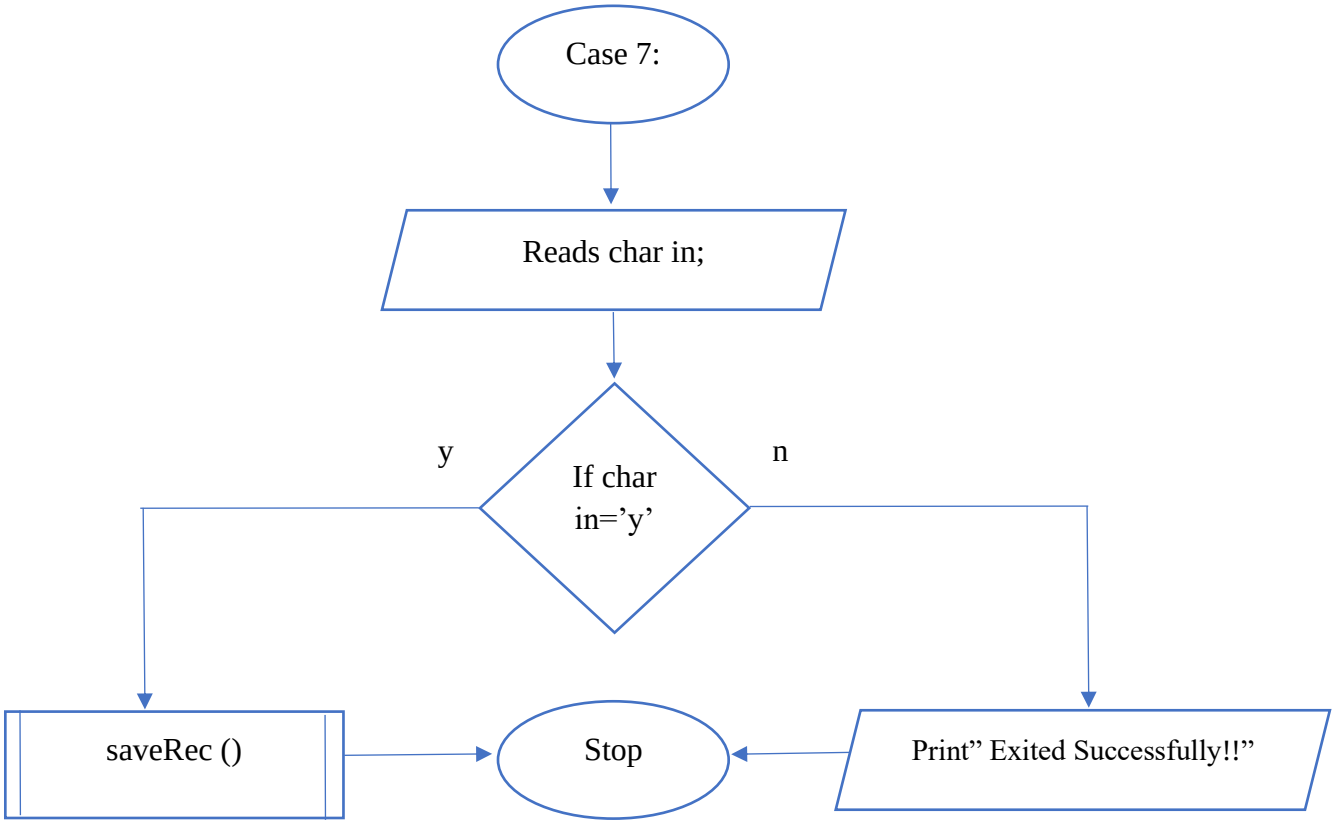

Case 2:
Reads the Input
(Strcmp (firstCap(attr), pr->phoneRec[i]->fName) == 0
strcmp(firstCap(attr), pr->phoneRec[i]->lName) == 0
strcmp (attr, pr->phoneRec[i]->pNum) == 0)
Yes
If ==0
No
A
B


A
Print Output
B
Print “Record not
Case 3:
Reads the Input
(Strcmp (firstCap(attr), pr->phoneRec[i]->fName) == 0
strcmp(firstCap(attr), pr->phoneRec[i]->lName) == 0
strcmp (attr, pr->phoneRec[i]->pNum) == 0)
Yes
If ==0
No
Prints “” Deleted
Successfully”
Prints “No match
found in record to


Case 4:
Opens File in Read Mode
Reads and stores Data from the file
Closes the file


Case 5:
Opens File in Write Mode
Stores the Data entry in the Database
Closes the File


Case 6:
Reads as char arr
sortRec ()
Sort by First Name
Sort by Last Name
Sort by Phone Number


Case 7:
Reads char in;
y
n
If char in='y'
saveRec ()
Stop
Print" Exited Successfully!!"

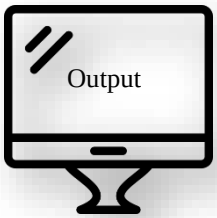
Features

- Secured:

  For the security of the telephone records, we have a separate login page where a user must first register their name and password, and only if those usernames and passwords match the data in the database during the login procedure will that user be granted access to the telephone directory.

- Print Records

  This feature displays the data that are stored in the Telephone directory Database. The data is tabulated in a way that distinguishes between first and last names, as well as phone numbers.



- Append

  This lets us to append/add the First name, Last name and Telephone number in our existing Database.

- Search

  This feature allows user to search any specific telephone number of a person by entering first name or last name of the person. Both names are presented on the output screen if there are two different entries with the same name.

- Delete

  This feature allows the user to remove any person's record from the directory. This can be done by typing the user's username or last name, or by entering the number of the order in which it is sorted if there are many users with the same name.

- Load

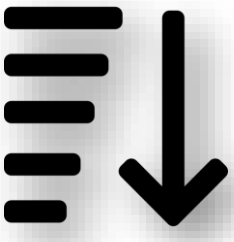
  The load is performed after any modifications to the file have been saved, because if it is performed before saving, the garbage value from the empty file may provide undesirable results.

- Save

  After appending, deleting, and sorting data, this function of the application is used to store the record in the Database. This should be done right after some modification is done.

- Sort

  This feature allows the user to sort the directory's records. The data can be sorted by first name, last name, and phone number.



- Exit

  This function aids the user in exiting the program. The user is asked if he or she wants to keep the changes made while exploring the program before departing it.

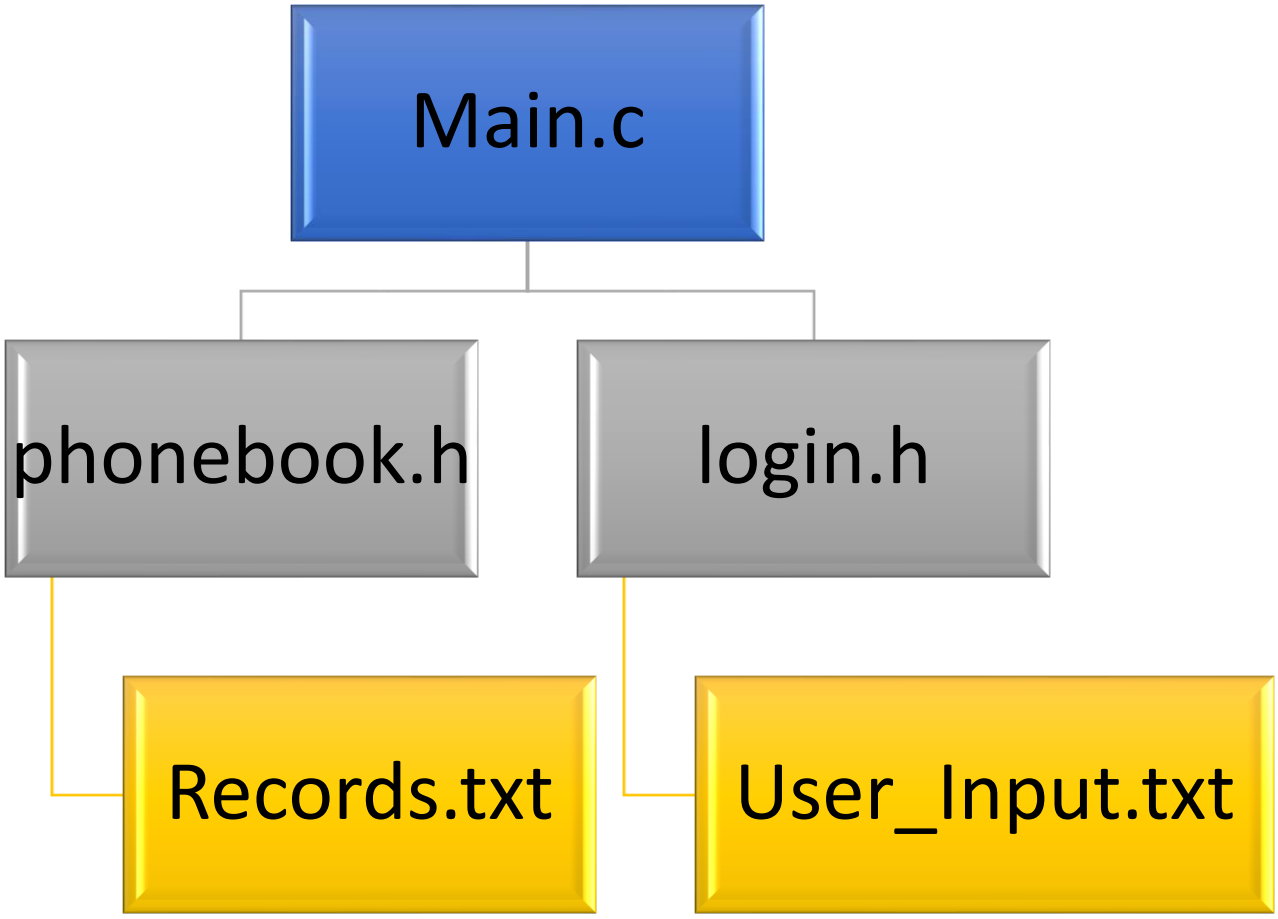
Conclusion with future enhancement

Using experimental scenarios and the language C, the application software was successfully implemented. Other functions of this application include making it simple to search, remove, update, and remember our information. examining other management systems that we described in the reference cases Our project necessitates crucial upgrades, which can be incorporated in the future or, to put it another way, future enhancements. Our program lacks double authentication, which could be a security flaw that will be rectified in future releases.

We were also unable to add several contacts to the same individual, but since everyone in today's society has many phone numbers, we planned to provide this capability in the future. We deal with "TELEPHONE DIRECTORY" in various forms in our daily lives. For example, we use the contact application on our phones. We can search, save, remove, adjust, and sort all of the data in the contact application on our phones, and our software does the same, so we can argue that our application, with a good user interface and other lacking functionalities as mentioned above, can be a replication of the contact application.

Code Architecture

The project comprises of a Main.c file, which has all of the primary modules, as well as two main header files, “phonebook.h” and “login.h,” which contain all of the functions required for the program to execute. Additionally, the login.h header file contains User Input.txt as its Database txt file, and the phonebook.h header file contains Records.txt as its Record tracking txt file/Database.



Source Code

1. Main.c

This is the source code for our main file, Main.c, which contains two key header files, namely "login.h" and "phonebook.h."

```
// ____________________________Header Files_____________________________
#include "login.h"
#include "phonebook.h"

// First Create a Blank User_Input.txt and Records.txt File in the same appli
cation Directory

// This system can store up to 20 number of users Data.

int main()
{

    system("color e");
    printf("\n\nLoading");
    for (int i = 0; i < 8; i++)
    {
        printf(".");
        Sleep(370);
    }
    FILE *fp = fopen(filename, "r");
    int i = 0, Num, Num1;
    User u;

    if (NULL == fp)
    {
        printf("\nFILE NOT FOUND");
        exit(1);
    }

    for (i = 0; i < USER_MAX; i++)
    {
        char User_Name[100];
        char User_Password[10];
        fscanf(fp, "%s%s", User_Name, User_Password);
        strcpy(list[i].name, User_Name);
        strcpy(list[i].password, User_Password);
    }

    int choice = Start_Options();

    switch (choice)
    {

        // This Case is for login

    case 1:
        system("cls");
        u = login ();
        if (1 == exist(u))
```

```
        {

            // More Main Codes to include.

            printf ("\n\n Loading");
            for (int i = 0; i < 8; i++)
            {
                printf (".");
                Sleep (370);
            }

            system("cls");

            printf ("\n\n\n\n\t\t\t\t\t\tHello
            , %s Welcome to the Program.", Confirmed_User.name);
            printf ("\n\n Loading");
            for (int i = 0; i < 8; i++)
            {
                printf (".");
                Sleep (250);
            }
            system("cls");
            PhoneRec pr = initPhoneRec ();
            loadRec(pr);
            printf ("Welcome to Our TELEPHONE DIRECTORY\n\n");
            mainMenu(pr);


        }
        else
        {

            Printf ("\nWRONG USERNAME OR PASS\n");
            printf ("\n 1 ==> Retry \n 0 ==> Exit\n\n   ==> ");
            scanf ("%d", &Num);
            switch (Num)
            {
            case 1:
                system("cls");
                main ();
                break;
            case 0:
                exit(0);
                break;
            default:
                printf("\nWrong Input Do you Want to Try again");
                printf("\n 1 ==> Retry \n 0 ==> Exit\n\n   ==> ");
                scanf("%d", &Num1);
                switch (Num1)
                {
                case 1:
                    system("cls");
                    main();
                    break;
                case 0:
                    exit(0);
                    break;
```

```
                default:
                    system("cls");
                    printf("\n\nAgain, the input is Wrong........");
                    main();
                    break;
                }
            }
        }
        break;

        // This Case is for Registration

    case 2:
        system("cls");
        registerUser();
        main ();
        break;

        // This Case is to Exit

    case 0:
        exit (0);
        break;

        // This Case is for the Default if any wrong key is Pressed except
        0, 1 and 2

    default:
        printf ("\n Wrong Input Do You Want to Try again");
        printf ("\n 1 ==> Retry \n 0 ==> Exit\n\n   ==> ");
        scanf ("%d", &Num);
        switch (Num)
        {

        case 1:
            system("cls");
            main ();
            break;
        case 0:
            exit (0);
            break;
        default:
            system("cls");
            printf ("\n\n Again, the input is Wrong..........");
            main ();
            break;

        }
    }
}
```

2. Login.h

This is the source code for the “login.h” header file, which contains the terminal login function. It is the first component of the application, in which a user must register by name before being able to login and access the "TELEPHONE DIRECTORY" services.

```
//_____________________________ Header Files _____________________________


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#define USER_MAX 20

// Declaration of Global Variable

char User_Name[100];

//Defining Structure having Name and Password as its Member

typedef struct
{

    char name[100];
    char password[10];

} User;

// define a global array, type User, size is USER_MAX

User list[USER_MAX];
User Confirmed_User;


// user.txt needs to br created your application directory before Running the
 prgram

char *filename = "User_Input.txt";


//_____________________________Function Starts_____________________________

// Login Function

User login()
{

    char name[100];
    char password[10];
    printf("\nEnter name:");
    scanf("%s", name);
```

```
    strcpy(Confirmed_User.name, name);

    printf("\nEnter password:");
    scanf("%s", password);

    strcpy(Confirmed_User.password, password);

    return Confirmed_User;

}

// Funcitn for Appending the name and Password of the user

void writeToFile(User u)
{

    FILE *fw = fopen(filename, "a+");
    fprintf(fw, u.name);
    fprintf(fw, "\t\t");
    fprintf(fw, u.password);
    fprintf(fw, "\n");
    fclose(fw);

}

// Funciton to determine whether the user exists

int exist(User u)
{

    int i;
    for (i = 0; i < USER_MAX; i++)
    {
        if (0 == strcmp(list[i].name, u.name) && 0 == strcmp
        (list[i].password, u.password))
        {
            return 1;
        }
    }
    return -1;

}

//Funciton to Register the user

void registerUser()
{

    char name[100];
    char password[10];
    User user;
    int i;

    printf("\nEnter your name:");
    scanf("%s", name);
    strcpy(user.name, name);
```

```c
    // determine whether there has been a registered user

    for (i = 0; i < USER_MAX; i++)
    {

        if (strcmp(list[i].name, name) == 0)
        {
            printf("\nUserName Already Taken.\n");
            printf("\nRegistration Failed :(\n");
            return;

        }
    }


    printf("\nEnter your password:");
    scanf("%s", password);

    printf("\nRegistration Successful :)\n");
    strcpy(user.password, password);

    writeToFile(user);

}

// Function for Showing the Options at First

int Start_Options()
{

    system("cls");
    int choice;
    printf("\n1==> Login\n");
    printf("2==> Register\n");
    printf("0==> Exit\n");
    printf("\nEnter your Choice: ");
    scanf("%d", &choice);
    return choice;

}
```

3. Phonebook.h

The “phonebook.h” header file contains all of the directory's major functionalities, such as sorting, appending, displaying, removing, and searching.

```
#include <ctype.h>
#define MaxR 5000
#define MaxNL 20

// ____________________________________STRUCTS____________________________________

typedef struct
{
    char fName[MaxNL + 1];
    char lName[MaxNL + 1];
    char pNum[11];
} RType, *Record;

typedef struct
{
    Record phoneRec[MaxR];
    int recIndex;
} PRType, *PhoneRec;

void mainMenu();
void sort(PhoneRec pr);
void addRecord(PhoneRec pr);
void load(PhoneRec pr);

// ___________________________________FUNCTIONS___________________________________

// creates an empty collection of records
PhoneRec initPhoneRec(void)
{
    PhoneRec pr = (PhoneRec)malloc(sizeof(PRType));
    pr->recIndex = 0;
    return pr;
}

//Goes through array and return 1 if phone number found and 0 if not.
int checkDupNum(PhoneRec pr, char *input)
{

    for (int i = 0; i < pr->recIndex; i++)
    {
        if (strcmp(input, pr->phoneRec[i]->pNum) == 0)
        {
            return 1;
        }
    }
    return 0;
}

// sort: display all records in ascending order based on a particular field
void swap(Record list[], int i, int j)
{
    //swap list[i] and list[j]
    Record hold = list[i];
    list[i] = list[j];
```

```
    list[j] = hold;
} //end swap

//splitting part of quicksort
int partition(Record A[], int lo, int hi, char ans)
{
    //partition A[lo] to A[hi] using A[lo] as the pivot
    Record pivot = A[lo];
    int lastSmall = lo;
    int cmp = 0;
    for (int h = lo + 1; h <= hi; h++)
    {
        if (ans == 'f')
            cmp = strcmp(A[h]->fName, pivot->fName);
        else if (ans == 'l')
            cmp = strcmp(A[h]->lName, pivot->lName);
        else
            cmp = strcmp(A[h]->pNum, pivot->pNum);

        if (cmp < 0)
        {
            ++lastSmall;
            swap(A, lastSmall, h);
        }
    }
    swap(A, lo, lastSmall);
    return lastSmall; //return the division point
} //end partition

//Divide and conquer, recursive algorithm, which calls partition
void quicksort(Record A[], int lo, int hi, char ans)
{
    //sorts A[lo] to A[hi] in ascending order
    if (lo < hi)
    {
        int dp = partition(A, lo, hi, ans);
        quicksort(A, lo, dp - 1, ans);
        quicksort(A, dp + 1, hi, ans);
    }
} //end quicksort

// capitalize first word
char *firstCap(char *s)
{
    int length = strlen(s);

    for (int i = 0; i < length; i++)
    {
        if (isalpha(s[0]))
        {
            s[0] = toupper(s[0]);
        }
        if (isalpha(s[i]) && i != 0)
        {
            s[i] = tolower(s[i]);
        }
    }

    return s;
}

// print all records in phone book
int cleanInput(void)
```

```
{
    while (getchar() != '\n')
        ;
    return 1;
}

void printRec(PhoneRec pr)
{
    printf("  NO.          FIRST NAME          LAST NAME          PHONE NUMBER\n");
    printf("_____________________________________________________________________\n\n")
;
    for (int i = 0; i < pr->recIndex; i++)
    {

        printf("%4d.%20s%19s%22s\n", i + 1, pr->phoneRec[i]->fName,
        pr->phoneRec[i]->lName,
        pr->phoneRec[i]->pNum);
    }
    printf("\n");
}

// add a new record
void addRec(PhoneRec pr, Record r)
{
    if (!checkDupNum(pr, r->pNum))
    {
        pr->phoneRec[pr->recIndex] = r;
        pr->recIndex++;
    }
}

// creates an empty record and returns the record struct
Record initRec(char *fName, char *lName, char *pNum)
{
    Record r = (Record)malloc(sizeof(RType));
    strncpy(r->fName, fName, MaxNL);
    r->fName[MaxNL] = '\0';
    strncpy(r->lName, lName, MaxNL);
    r->lName[MaxNL] = '\0';
    strncpy(r->pNum, pNum, 10);
    r->pNum[MaxNL] = '\0';
    return r;
}

// retrieve: display a record that contains a particular attribute value
void retrieveRec(PhoneRec pr)
{
    char *attr = (char *)malloc(MaxNL + 1);
    int found = 0;
    int count = 1;
    printf("\nEnter first name, last name, or phone number to look for:\n");
    scanf("%s", attr);

    printf("  NO.          FIRST NAME          LAST NAME          PHONE NUMBER\n");
    printf("_____________________________________________________________________\n\n")
;

    for (int i = 0; i < pr->recIndex; i++)
    {
        if (strcmp(firstCap(attr), pr->phoneRec[i]->fName) == 0 ||
            strcmp(firstCap(attr), pr->phoneRec[i]->lName) == 0 ||
            strcmp(attr, pr->phoneRec[i]->pNum) == 0)
        {
```

```
            printf("%4d.%20s%19s%22s\n", count, pr->phoneRec[i]->fName,
            pr->phoneRec[i]->lName,
            pr->phoneRec[i]->pNum);
            count++;
            found = 1;
        }
    }

    if (found == 0)
    {
        printf("\nRecord not found\n");
    }

    printf("\n");
}

// delete: delete an existing record given a specific attribute of that record
void deleteRec(PhoneRec pr)
{
    int found = 0;
    char *attr = (char *)malloc(MaxNL + 1);
    int count = 0;

    printf("\nEnter first name, last name, or phone number to delete:\n");
    scanf("%s", attr);
    printf("\n");

    printf("  NO.          FIRST NAME          LAST NAME          PHONE NUMBER\n");
    printf("_____________________________________________________________________\n\n")
;

    for (int i = 0; i < pr->recIndex; i++)
    {
        if (strcmp(firstCap(attr), pr->phoneRec[i]->fName) == 0 ||
            strcmp(firstCap(attr), pr->phoneRec[i]->lName) == 0 ||
            strcmp(attr, pr->phoneRec[i]->pNum) == 0)
        {
            printf("%4d.%20s%19s%22s\n", count + 1, pr->phoneRec[i]->fName,
            pr->phoneRec[i]->lName,
            pr->phoneRec[i]->pNum);
            count++;
        }
    }

    int recNum = 1;
    char newLine;
    if (count > 1)
    {
        do
        {
            printf("\nWhich record do you want to to delete from the above list?\n");
        } while (((scanf("%d%c", &recNum, &newLine) != 2 || newLine != '\n') &&
                 cleanInput()) ||
                recNum < 1 || recNum > count);
    }

    for (int i = 0; i < pr->recIndex; i++)
    {

        if (strcmp(attr, pr->phoneRec[i]->fName) == 0 ||
            strcmp(attr, pr->phoneRec[i]->lName) == 0 ||
            strcmp(attr, pr->phoneRec[i]->pNum) == 0)
        {
```

```c
            found = 1;
            if (recNum == 1)
            {
                // move all elements right of index to left by one position
                for (int j = i; j < pr->recIndex - 1; j++)
                {
                    pr->phoneRec[j] = pr->phoneRec[j + 1];
                }
                pr->recIndex--;
                printf("\nThis record was successfully deleted.\n\n");
            }
            else
                recNum--;
        }
    }

    if (found == 0)
        printf("\nNo record match found to delete.\n\n");

    //printRec(pr);
}

// load: read data from a file which contains data of the phone book
void loadRec(PhoneRec pr)
{
    char fName[MaxNL + 1];
    char lName[MaxNL + 1];
    char pNum[11];
    FILE *rFile = fopen("Records.txt", "r"); // opens a file for reading

    if (rFile == NULL) // file does not exist
        printf("File not found.\n\n");
    else
    {
        while (!feof(rFile))
        {
            fscanf(rFile, "%s %s %s", fName, lName, pNum);

            if (feof(rFile))
            {
                break;
            }
            else
            {
                Record r = initRec(fName, lName, pNum);

                addRec(pr, r);
            }
        }
    }

    printf("\n");
    quicksort(pr->phoneRec, 0, pr->recIndex - 1, 'f');
    fclose(rFile);
}

// save: write data of phone book to a file
void saveRec(PhoneRec pr)
{
    FILE *wFile = fopen("Records.txt", "w"); // opens a new file for writing

    if (wFile == NULL)
    {
```

```
        printf("File not found.\n\n");
    }

    for (int i = 0; i < pr->recIndex; i++)
    {
        fprintf(wFile, "%s %s %s\n", pr->phoneRec[i]->fName, pr->phoneRec[i]-
>lName, pr->phoneRec[i]->pNum);
    }

    fclose(wFile);
}
//Function for checking if string contains anything other than characters
int isChar(char *input)
{
    int len = strlen(input);

    for (int i = 0; i < len; i++)
    {
        if (!isalpha(input[i]))
        {
            printf("Must be characters only\n");
            return 0;
        }
    }
    return 1;
}
//Function for checking if string contains anything other than numbers
int isNum(char *input)
{
    int len = strlen(input);

    for (int i = 0; i < len; i++)
    {
        if (isalpha(input[i]))
        {
            return 0;
        }
    }
    return 1;
}
void mainMenu(PhoneRec pr)
{
    int input;
    printf("0. Print Phone Records\n");
    printf("1. Add Record\n");
    printf("2. Retrieve Record\n");
    printf("3. Delete Record\n");
    printf("4. Load File\n");
    printf("5. Save To File\n");
    printf("6. Sort\n");
    printf("7. Exit\n\n");
    do
    {
        printf("Choose an option: \n");
        scanf("%d", &input);
    } while ((input < 0 || input > 7) && cleanInput());

    if (input == 0)
    {
        printf("\n\n\n\nLoading");
        for (int i = 0; i < 8; i++)
        {
            printf("\n.");
```

```
            Sleep(150);
        }
        printf("\n\n\n");
        printRec(pr);
        mainMenu(pr);
    }
    else if (input == 1)
    {
        addRecord(pr);
        mainMenu(pr);
    }
    else if (input == 2)
    {
        retrieveRec(pr);
        mainMenu(pr);
    }
    else if (input == 3)
    {
        printf("\n");
        deleteRec(pr);
        mainMenu(pr);
    }
    else if (input == 4)
    {
        load(pr);
        mainMenu(pr);
    }
    else if (input == 5)
    {
        saveRec(pr);
        printf("\nState saved!\n\n");
        mainMenu(pr);
    }
    else if (input == 6)
    {
        sort(pr);
        printf("\n\n");
        mainMenu(pr);
    }
    else if (input == 7)
    {
        char in;
        do
        {
            printf("\nDo you want to save current state(y/n): ");
            scanf("%s", &in);

            if (in == 'y')
            {
                saveRec(pr);
                printf("State saved!\n");
                exit(0);
            }
            else if (in == 'n')
            {
                printf("\n Exicted Successfully !!! \n");
                exit(0);
            }
            else
                printf("Invalid Input!\n");

        } while (in);
    }
```

```c
}

//Adds user input into records which is added to the PhoneRec array.
void addRecord(PhoneRec pr)
{

    char first[MaxNL + 1];
    char last[MaxNL + 1];
    char pnum[11];
    char in = 'y';

    while (in != 'n')
    {
        do
        {
            printf("\nEnter First Name: ");
            scanf("%s", first);
        } while (!isChar(first));

        do
        {
            printf("Enter Last Name: ");
            scanf("%s", last);
        } while (!isChar(last));
        do
        {
            printf("Enter Phone Number: ");
            scanf("%s", pnum);
            if (!isNum(pnum))
                printf("\nMust be Numbers only\n\n");
            else if (strlen(pnum) != 10)
                printf("\nPhone number must be 10 numbers long.\n\n");
            else if (checkDupNum(pr, pnum))
                printf("\nPhone number already exists.\n\n");
        } while (checkDupNum(pr, pnum) || !isNum(pnum) || strlen(pnum) != 10);

        Record r = initRec(firstCap(first), firstCap(last), pnum);
        addRec(pr, r);
        quicksort(pr->phoneRec, 0, pr->recIndex - 1, 'f');
        do
        {
            printf("\nDo you want to add another record? (y/n)");
            scanf("%s", &in);
            if (in == 'y')
                break;
            else if (in == 'n')
            {
                printf("\n");
                mainMenu(pr);
            }
            else
                printf("Enter valid input\n\n");
        } while (in);
    }
}

//Prompts user to choose a sort criteria. Calls quicksort accordingly.
void sort(PhoneRec pr)
{
    char in[11];
    do
    {
        printf("\nSort by First Name(f), Last Name(l), or Phone Number(p)?\n");
```

```
        scanf("%s", in);
        printf("\n");
        firstCap(in);
        if (!strcmp(in, "F"))
        {
            quicksort(pr->phoneRec, 0, pr->recIndex - 1, 'f');
            printf("\n\nLoading");
            for (int i = 0; i < 8; i++)
            {
                printf("\n.");
                Sleep(150);
            }
            printf("\n\n");
            printRec(pr);
            printf("Sorted by First Name\n\n");
            mainMenu(pr);
        }

        else if (!strcmp(in, "L"))
        {
            quicksort(pr->phoneRec, 0, pr->recIndex - 1, 'l');
            printf("\n\nLoading");
            for (int i = 0; i < 8; i++)
            {
                printf("\n.");
                Sleep(150);
            }
            printf("\n\n");
            printRec(pr);
            printf("Sorted by Last Name\n\n");
            mainMenu(pr);
        }
        else if (!strcmp(in, "P"))
        {
            quicksort(pr->phoneRec, 0, pr->recIndex - 1, 'p');
            printf("\n\nLoading");
            for (int i = 0; i < 8; i++)
            {
                printf("\n.");
                Sleep(150);
            }
            printf("\n\n");
            printRec(pr);
            printf("Sorted by Phone Number\n\n");
            mainMenu(pr);
        }
        else
        {
            printf("Invalid input\n");
        }
    } while ((strcmp(in, "F") || strcmp(in, "L") || strcmp(in, "P")));
}
//Calls loadRec and prints success statement.
void load(PhoneRec pr)
{

    loadRec(pr);
    printf("Load Successful\n\n");
}
```

Output Snippets

To begin, the application displays three alternatives for you to choose from: login, register, or exit the program. Because you haven't registered your user name yet, choose option 2 and register your username and password.

```
main1.c   login.h   phonebook.h   records.txt   User_Input.txt

User_Input.txt
1

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

1--> Login
2--> Register
0--> Exit

Enter your Choice: 2
```

You'll now be led to option ‘2', where you'll be requested to enter your name and password, as well as a valid input, and the program will save the data you've entered in the login system's database.

```
main1.c   login.h   phonebook.h   records.txt   User_Input.txt

User_Input.txt
1   nabin      8080
2

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Enter your name:nabin

Enter your password:8080

Registration Successful :)

Loading....
```

Since the system's database has your identity. You can now select option 1 and proceed to the login page, where you will input the same username and password, allowing you to continue with the program.

```
main1.c    login.h    phonebook.h    records.txt    User_Input.txt
User_Input.txt
1    nabin        8080
2

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Enter Username : nabin

Enter Password : 8080

Loading......
```

The system now double-checks the user name entered by the user to see if it exists in the database. If the user's name is found in the database, he or she is granted access to the "TELEPHONE DIRECTORY".

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                    Code

                    Hello,nabin Welcome to the Program.

Loading.......
```

If, instead of entering the same username and password as in the database, the user enters the wrong user name and password, an error notice appears with the choice to retry or exit.

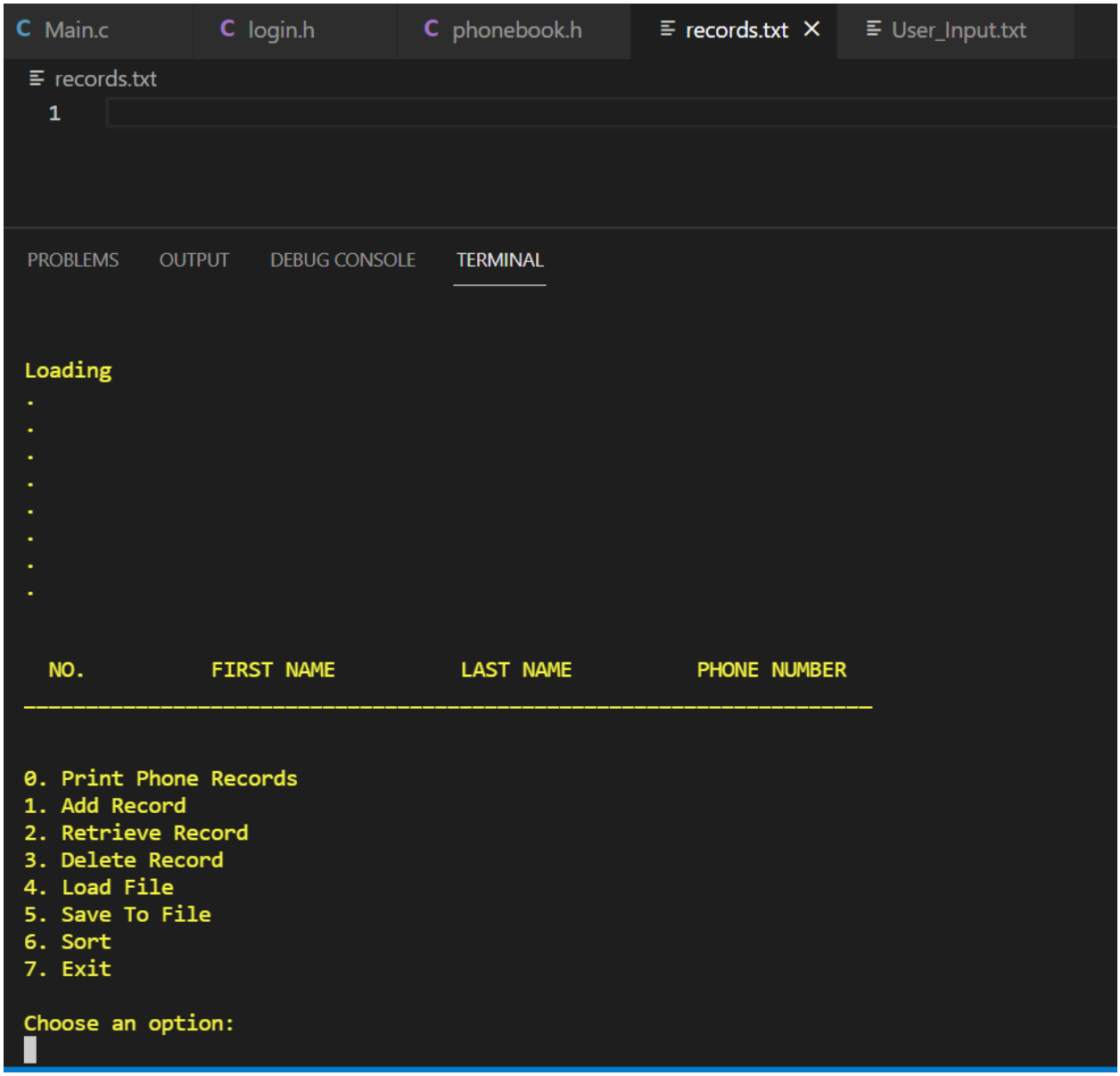
```
Main.c    login.h    phonebook.h    records.txt    User_Input.txt

User_Input.txt
1   nabin       8080
2

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Enter Username : fdssk

Enter Password : 4454

WRONG USERNAME OR PASS

 1 --> Retry
 0 --> Exit

   Your choice -->
```

Now that the login process has been completed, the main part of the Program will begin, which will allow the user to enjoy the various functions of the program.

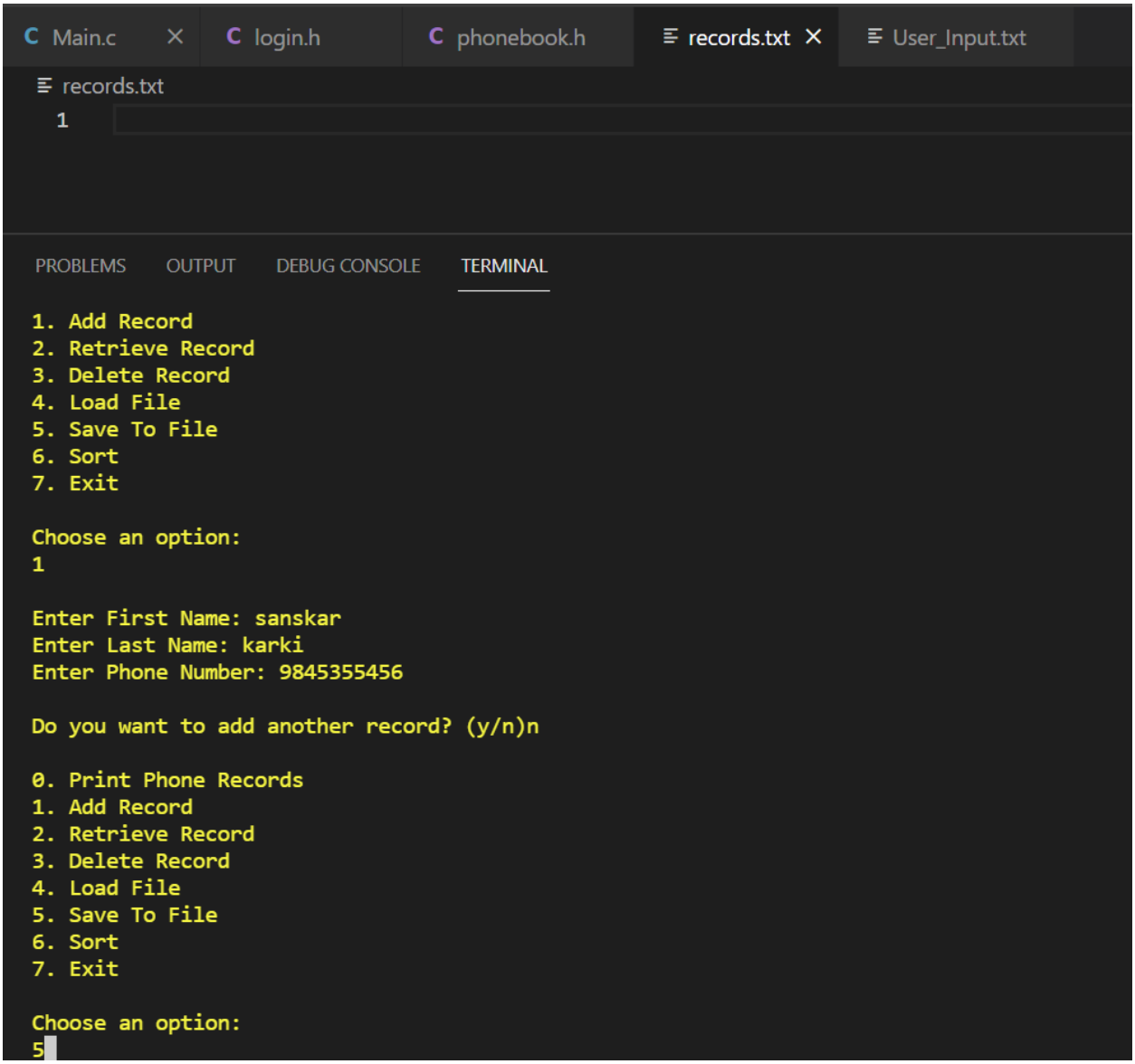
After completing the login part and gaining access to the telephone directory, the user is presented with a variety of options from which to choose.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Welcome to Our TELEPHONE DIRECTORY

0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
```

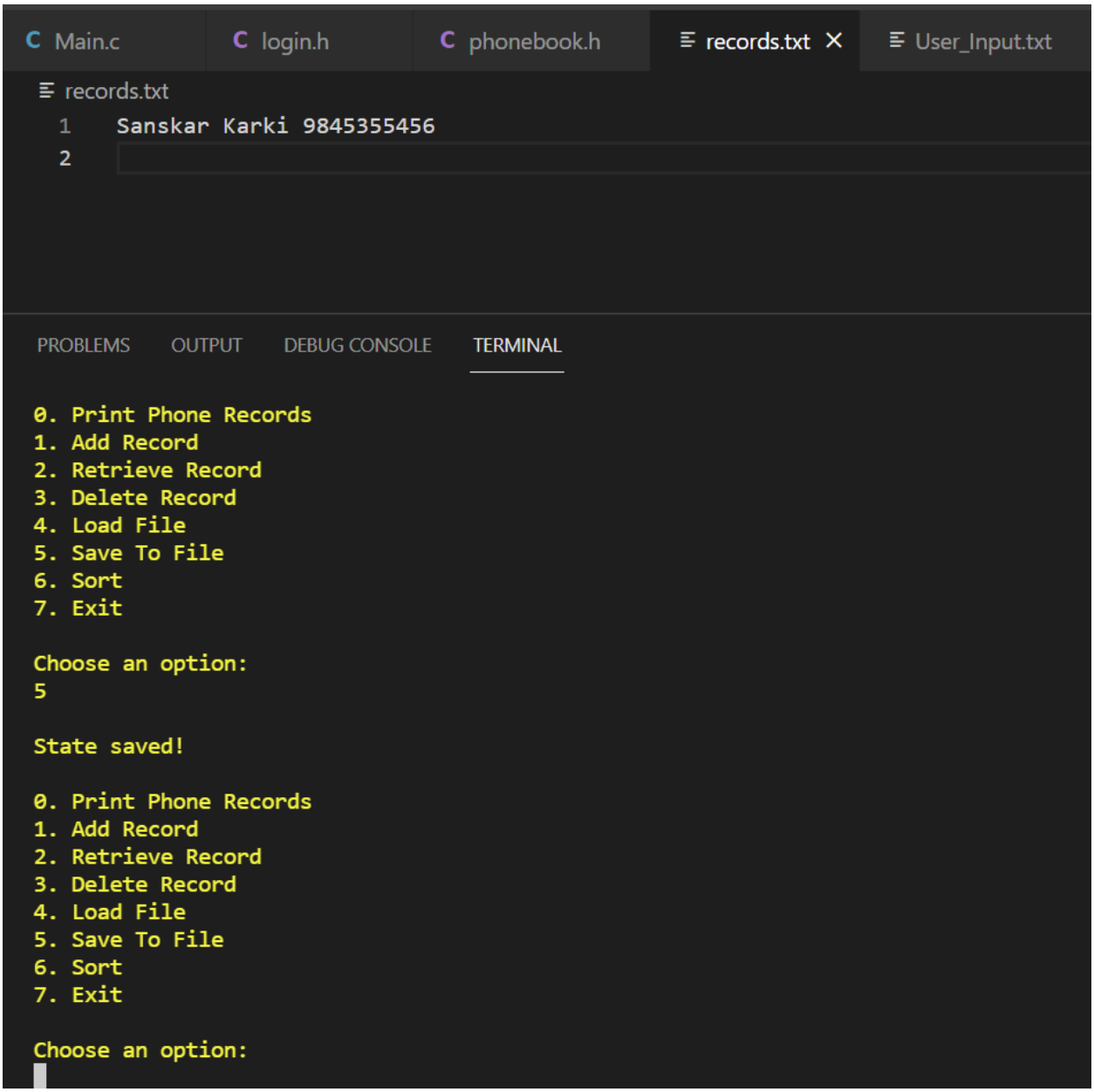
Now, because the file is empty initially, if the user selects option 0 to print the record, nothing will appear because the file is blank.



Now, a user can add data to the file by selecting the add option, which prompts the user to provide their first, last, and phone numbers.



To save the data that the user has entered, the user should save it to a file using the save to file option, which pushes the data to the database and stores it.



Following a series of multiple data entries. If a user wants to access some specific information, he or she can do so by selecting the search option and searching by first name, last name, or phone number.

Main.c | login.h | phonebook.h | records.txt | User_Input.txt

records.txt

```
Nabin Khatri 9869547452
Sanskar Karki 9845355456
Sanskar Singh 9845632107
Shishir Kafle 9852036147
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
2

Enter first name, last name, or phone number to look for:
sanskar
  NO.          FIRST NAME          LAST NAME          PHONE NUMBER
-----------------------------------------------------------------------

   1.             Sanskar             Karki             9845355456
   2.             Sanskar             Singh             9845632107

0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
3
```

If the user wants to erase the data, he or she can do so now. The user must select the option to delete the record. Here, too the search feature is used and the user is asked to enter the first name, last name, or phone number of the data we want to remove.

When you search for data by first name or last name, if there are two people with the same name, it displays both of their records and asks which one you wish to delete. And the user can simply erase data by inputting the number in which it exists.

```
Main.c    login.h    phonebook.h    records.txt ×    User_Input.txt

records.txt
1   Nabin Khatri 9869547452
2   Sanskar Karki 9845355456
3   Sanskar Singh 9845632107
4   Shishir Kafle 9852036147
5

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

   1.            Sanskar            Karki            9845355456
   2.            Sanskar            Singh            9845632107

0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
3


Enter first name, last name, or phone number to delete:
sanskar

  NO.          FIRST NAME          LAST NAME          PHONE NUMBER
-----------------------------------------------------------------------

   1.            Sanskar            Karki            9845355456
   2.            Sanskar            Singh            9845632107

Which record do you want to to delete from the above list?
2
```

After the data has been deleted, the user can utilize the save to file feature to update the data in the database.

Main.c | login.h | phonebook.h | records.txt | User_Input.txt

records.txt

```
Nabin Khatri 9869547452
Sanskar Karki 9845355456
Shishir Kafle 9852036147
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
This record was successfully deleted.

0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
5

State saved!

0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
6
```

Another important aspect of the program is the user's ability to sort the data in the datable. The data can be sorted using three distinct methods: first name, last name, and phone number.

```
0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
6

Sort by First Name(f), Last Name(l), or Phone Number(p)?
f
```

The following is how the data would be sorted if it were sorted by first name:

```
Sort by First Name(f), Last Name(l), or Phone Number(p)?
f



Loading
.
.
.
.
.
.
.
.

  NO.          FIRST NAME          LAST NAME          PHONE NUMBER
-------------------------------------------------------------------

   1.                Nabin             Khatri            9869547452
   2.              Sanskar              Karki            9845355456
   3.              Shishir              Kafle            9852036147

Sorted by First Name
```

The following is how the data would be sorted if it were sorted by last name:

```
  NO.          FIRST NAME          LAST NAME          PHONE NUMBER
-------------------------------------------------------------------

   1.              Shishir              Kafle            9852036147
   2.              Sanskar              Karki            9845355456
   3.                Nabin             Khatri            9869547452

Sorted by Last Name
```

The following is how the data would be sorted if it were sorted by number:

```
  NO.          FIRST NAME          LAST NAME          PHONE NUMBER
-----------------------------------------------------------------------

   1.             Sanskar              Karki            9845355456
   2.             Shishir              Kafle            9852036147
   3.               Nabin             Khatri            9869547452

Sorted by Phone Number
```

Finally, if the user wants to exit the program, he or she can do so by selecting the exit option, which asks whether the user wants to save the last data or not. If the user selects ‘y’, the data is saved; if the user selects ‘n’, the modification made while the user was using the program is not saved.

```
0. Print Phone Records
1. Add Record
2. Retrieve Record
3. Delete Record
4. Load File
5. Save To File
6. Sort
7. Exit

Choose an option:
7

Do you want to save current state(y/n): y
State saved!
PS C:\Users\nabin\Desktop\Final_project>
```

Refrence

[1] https://stackoverflow.com/

[2] https://www.w3schools.com/

[3] https://www.javatpoint.com/c-string-functions

[4] https://www.quora.com/

[5] https://www.programiz.com/c-programming/c-structures-pointers

[6] https://www.flaticon.com/