A Mini Project Report

On

**PHONEBOOK APPLICATION**

By

**AKASH.S.VORA**

**1602-19-733-126**

**AKHIL KUMBAM**

**1602-19-733-127**



**Department of Computer Science & Engineering**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2020 - 21**

**ACKNOWLEDGEMENT**

With immense pleasure, we record our deep sense of gratitude to our guide Dr. D. Baswaraj, Professor, Vasavi College of Engineering, Hyderabad, for the valuable guidance and suggestions, keen interest and thorough encouragement extended throughout the period of the project work. I consider myself lucky enough to be part of this project. This project would add as an asset to my academic profile.

We express our thanks to all those who contributed for the successful completion of our project work.

**TABLE OF CONTENTS**

**Page No.**

1. **Introduction …………………………………………………. 1**
2. **Abstract …………………………………………………….... 2**
3. **Objective …………………………………………................... 2**
4. **System Requirements .…………………................................. 3**
5. **Functionalities……………………………………………….... 3**
6. **Data Structures Used ………………………………………… 4**
7. **Program ……..……………………………………………….. 7**
8. **Output of the Program ……………………………………… 27**
9. **Future Work ………………………………………………… 43**
10. **References ………………………………………………......... 43**

**INTRODUCTION**

Phone book system is a simple application that is designed for the user to store the details of their contacts. Rather than going through the pages of their diaries and copies to search for their contact details, a person can simply use this application to view any of his stored contacts. A user can also add or update or delete the contact details according to his need. This application uses the basic concepts of functions as well as data structures.

Adding new records , sorting them, modifying them and updating, search for saved contacts, and deleting the phonebook records are the basic functions which make up the main menu of this Phonebook application.

**ABSTRACT**

Phonebook system is a small application developed to manipulate the contact details. In olden days people used to store all important contact details in books and papers. Here we propose a new system, by using this application we can store all the contact details of a user at the same place.

Here the user is provided with the options like insert a new contact, search a saved contact, display the contacts, update and delete data from the already existing contacts etc.,

This system is convenient to all the users as we provide most of the features of a manual phonebook.

**OBJECTIVE**

To build an application which allows the users to perform operations such as insertion, deletion, display, sort and update the contacts details of the user.

**SYSTEM REQUIREMENTS**

**Hardware:**

* Minimum RAM required: 512 mb
* Input devices: Mouse, Keyboard
* Output devices: Monitor

**Software:**

* Code::Blocks IDE
* Windows 7 or above

**FUNCTIONALITIES**

* start() – To start the phonebook application by prompting the user to enter the first entry into the phonebook.
* insert() – Insert an entry into the phonebook.
* search() – Search a record from your phonebook.
* search\_by\_char() – Search a record from your phonebook by character.
* update() – Update an entry in your phonebook.
* display() – Display all the records in your phonebook.
* sortList() – Sort all the phonebook records and display it in alphabetical order.
* delete\_data() – Delete a record from your phonebook.
* display\_rev() – Display all records of your phonebook in reverse oreder i.e in the order of the last inserted record to the first one.

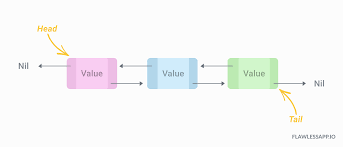
**DATA STRUCTURES USED**

1. **DOUBLY LINKED LIST (DLL) :**

In Computer Science, a **doubly linked list** is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains three fields: two link fields and one data field. The two link fields point to the previous and to the nextnode in the sequence of nodes and the data field contains the value.

The previous link of the head node and the next link of the tail node point to the NULL to make the traversal of the list easier

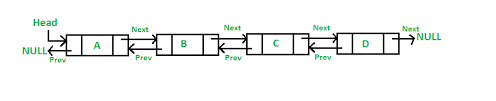
The two node links allow traversal of the list in either direction. While adding or removing a node in a doubly linked list requires changing more links than the same operations on a singly linked list, the operations are simpler and potentially more efficient (for nodes other than first nodes) because there is no need to keep track of the previous node during traversal or no need to traverse the list to find the previous node, so that its link can be modified.



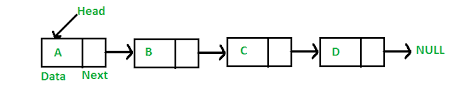
**ADVANTAGES OF DOUBLY LINKED LIST OVER SINGLY LINKED LIST :-**

1. A Doubly linked list can be traversed in both the forward and backward direction, whereas a Singly linked list can only traverse in one direction.
2. The delete operation in Doubly linked list is more efficient if the pointer to the node is given. In Singly linked list, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list needs to traversed which can be avoided in aDoubly linked list.
3. In Doubly linked list, We can quickly insert a new node before a given node using previous pointer which is not the case of Singly linked list.

A Doubly linked list :



A Singly linked list :



1. **ARRAYS :**

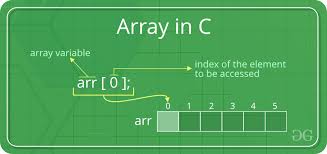
In Computer Science, an **array** is a data structure consisting of collections of elements, each identified by atleast one array index, stored at contiguous memory loacations. An array is stored such that the position of each elements can be computed from the index itself.

Arrays are among the oldest and most important data structures, and are used by almost every program. They are also used to implement many other data structures, such as lists and strings.

In most modern computers and many external storage devices, the memory is a one-dimensional array of words, whose indices are their addresses.

In our program, we have used arrays to accept the name of the user. In addition to this, we have also used character array to accept the name and phone number of the record that the user wants to add.

Array Representation :



**PROGRAM**

//Phonebook Application

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

struct phone{

struct phone\* prev;

char name[50];

char phonenum[20];

struct phone\* next;

};

struct phone\* head=NULL;

struct phone\* tail=NULL;

void start()

{

struct phone \*new;

new = (struct phone\*)malloc(sizeof(struct phone));

char n[50],p[20];

printf("Hello, You are now about to enter the first data into the phonebook!!!\n");

printf("Enter name: \n");

scanf("%s",n);

strcpy(new->name,n);

Inv1:

printf("Enter phone number: \n");

scanf("%s",p);

if(strlen(p)!=10)

{

printf("Invalid phone number! Please enter a valid number!\n");

goto Inv1;

}

strcpy(new->phonenum,p);

new->prev=NULL;

new->next=NULL;

head = new;

tail=new;

printf("\nCool.You have successfully added the first entry into your phonebook.\n");

printf("Try exploring more features!!\n\n");

}

void insert()

{

struct phone\* temp2= head;

struct phone \*new1;

new1 = (struct phone\*)malloc(sizeof(struct phone));

char n[50],p[20];

printf("\nYou are now about to INSERT an entry into the phonebook!!!\n");

printf("Enter name: \n");

scanf("%s",n);

strcpy(new1->name,n);

Inv:

printf("Enter phone number: \n");

scanf("%s",p);

if(strlen(p)!=10)

{

printf("Invalid phone number! Please enter a valid number!\n");

goto Inv;

}

strcpy(new1->phonenum,p);

if(head==NULL)

{

head = new1;

tail = new1;

}

else

{

while(temp2->next!=NULL)

{

temp2 = temp2->next;

}

new1->prev = temp2;

temp2->next = new1;

new1->next = NULL;

tail = new1;

}

printf("\nEntry added successfully!\n");

}

void search()

{

printf("Enter the name you want to search : ");

char search\_name[50];

scanf("%s",search\_name);

struct phone\* temp = head;

while(temp!=NULL)

{

if(strcmp(temp->name,search\_name)==0)

{

printf("Name - '%s' found in your phonebook!!\n",search\_name);

printf("DETAILS : \n");

printf("\nName : %s\t Phone number : %s\n",search\_name,temp->phonenum);

return;

}

temp=temp->next;

}

printf("Name - '%s' not found in your phonebook.\n",search\_name);

srch:

printf("Do you want to add the details?\t1.YES 2.NO\n");

int search\_var;

scanf("%d",&search\_var);

if(search\_var==1)

{

insert();

}

else if(search\_var==2)

{

return;

}

else

{

printf("Invalid option!");

goto srch;

}

}

void search\_by\_char()

{

char sname,buf,buf1;

scanf("%c",&buf);

printf("Enter the character with which you want to search names in your phonebook : ");

scanf("%c%c",&sname,&buf1);

struct phone\* temp = head;

printf("DETAILS : \n");

int flag=0;

while(temp!=NULL)

{

if(toupper(sname)==(temp->name)[0] || tolower(sname)==(temp->name)[0])

{

printf("\nName : %s\t Phone number : %s\n",temp->name,temp->phonenum);

flag=1;

}

temp=temp->next;

}

if(flag==0)

{

printf("\nNames beginning with '%c' not found in your phonebook!!\n",sname);

}

}

void update()

{

printf("Enter the name of the person whose details you wish to edit : ");

char upd\_name[50];

scanf("%s",upd\_name);

struct phone\* temp = head;

while(temp!=NULL)

{

if(strcmp(temp->name,upd\_name)==0)

{

printf("Name - '%s' found in your phonebook!!\n",upd\_name);

printf("Enter the details to update the existing record : \n");

printf("Enter name : ");

char upd[50],upd\_pnum[20];

scanf("%s",upd);

strcpy(temp->name,upd);

updpoint:

printf("Enter phone number : ");

scanf("%s",upd\_pnum);

if(strlen(upd\_pnum)!=10)

{

goto updpoint;

}

strcpy(temp->phonenum,upd\_pnum);

printf("\n\nDETAILS UPDATED SUCCESSFULLY!!\n");

printf("\nUPDATED DETAILS : \n");

printf("\nName : %s\t Phone number : %s\n",temp->name,temp->phonenum);

return;

}

temp=temp->next;

}

printf("Name - '%s' not found in your phonebook.\n",upd\_name);

updt:

printf("Do you want to add the details?\t1.YES 2.NO\n");

int update\_var;

scanf("%d",&update\_var);

if(update\_var==1)

{

insert();

}

else if(update\_var==2)

{

return;

}

else

{

printf("Invalid option!");

goto updt;

}

}

void display()

{

struct phone\* temp;

temp = head;

if(head == NULL)

{

printf("Your phonebook is empty at the moment. Start adding entries soon!!\n");

return;

}

printf("\nThe contents of your directory are:\n");

printf("Sno.\tName\tPhone number\n");

int i=1;

while(temp!=NULL)

{

printf("%d\t%s\t%s\n",i,temp->name,temp->phonenum);

i++;

temp = temp->next;

}

}

struct phone\* CopyList(struct phone\* head)

{

struct phone\* current = head;

struct phone\* newList = NULL;

struct phone\* tail = NULL;

while (current != NULL)

{

if (newList == NULL)

{

newList = (struct phone\*)malloc(sizeof(struct phone));

strcpy(newList->name,current->name);

strcpy(newList->phonenum,current->phonenum);

newList->next = NULL;

newList->prev = NULL;

tail = newList;

}

else

{

tail->next = (struct phone\*)malloc(sizeof(struct phone));

tail->next->prev = tail;

tail = tail->next;

strcpy(tail->name,current->name);

strcpy(tail->phonenum,current->phonenum);

tail->next = NULL;

}

current = current->next;

}

return newList;

}

void sortList() {

char temp[50];

char temp1[20];

if(head==NULL)

{

printf("Your phonebook is empty at the moment. Start adding entries soon!!\n");

return;

}

struct phone\* present = CopyList(head);

struct phone\* present1 = present;

struct phone\* index = NULL;

for(; present->next != NULL; present = present->next) {

for(index = present->next; index != NULL; index = index->next) {

if(strcmp(present->name,index->name)>0) {

strcpy(temp,present->name);

strcpy(present->name,index->name);

strcpy(index->name,temp);

strcpy(temp1,present->phonenum);

strcpy(present->phonenum,index->phonenum);

strcpy(index->phonenum,temp1);

}

}

}

printf("\nThe contents of your directory in alphabetical order are:\n");

printf("Sno.\tName\tPhone number\n");

int i=1;

while(present1!=NULL)

{

printf("%d\t%s\t%s\n",i,present1->name,present1->phonenum);

i++;

present1 = present1->next;

}

}

void delete\_data(char name[]) {

int pos = 0;

struct phone \*pre\_node;

if(head==NULL) {

printf("Phonebook is empty now!!\n");

return;

}

if(strcmp(head->name,name)==0)

{

if(head->next != NULL) {

head->next->prev = NULL;

head = head->next;

return;

} else {

head = NULL;

printf("\nRECORD DELETED SUCCESSFULLY!!\n");

printf("\nPhonebook is empty now!!\n");

return;

}

}

else if(strcmp(head->name,name)!=0 && head->next == NULL) {

printf("\nName - '%s' not found in the Phonebook!!\n",name);

return;

}

struct phone\* current = head;

while(current->next != NULL && strcmp(current->name,name)!=0 ) {

pre\_node = current;

current = current->next;

}

if(strcmp(current->name,name)==0)

{

pre\_node->next = pre\_node->next->next;

if(pre\_node->next != NULL)

{

pre\_node->next->prev = pre\_node;

}

else

tail = pre\_node;

free(current);

printf("\nRECORD DELETED SUCCESSFULLY!!\n");

}

else

printf("Name - '%s' not found in your phonebook!!\n",name);

}

void display\_rev()

{

struct phone\* temp;

temp = tail;

if(head == NULL)

{

printf("Your phonebook is empty at the moment. Start adding entries soon!!\n");

return;

}

printf("\nThe contents of your directory are:\n");

printf("Sno.\tName\tPhone number\n");

int i=1;

while(temp!=NULL)

{

printf("%d\t%s\t%s\n",i,temp->name,temp->phonenum);

i++;

temp = temp->prev;

}

}

//Driver Code

int main()

{

/\*

PHONEBOOK APPLICATION FEATURES

1.start()

2.insert()

3.delete()

4.search()

5.search\_by\_char()

6.update()

7.sort()

8.display()

9.display\_rev()

10.EXIT

\*/

printf("\*\*\*\*\*\*\*\*PHONEBOOK APPLICATION\*\*\*\*\*\*\*\*\n\n");

printf("Hello User! Please enter your name: ");

char uname[50];

gets(uname);

for (int i = 0; uname[i]!='\0'; i++) {

if(uname[i] >= 'a' && uname[i] <= 'z') {

uname[i] = uname[i] -32;

}

}

printf("\nWELCOME %s TO OUR ONLINE PHONEBOOK\t\t\n\n",uname);

int var;

start();

do{

printf("\nOUR PHONEBOOK FEATURES:\n");

printf("1. Insert contact into your phonebook.\n");

printf("2. Delete contact from your phonebook.\n");

printf("3. Display all contacts from your phonebook.\n");

printf("4. Display all contacts from your phonebook from last inserted record to the first inserted one.\n");

printf("5. Update a record of your phonebook.\n");

printf("6. Search a record from your phonebook.\n");

printf("7. Search a record from your phonebook using Character Search.\n");

printf("8. Display all contacts of your phonebook in alphabetical order.\n");

printf("9. EXIT\n");

printf("\nEnter the number for the corresponding operation : ");

scanf("%d",&var);

switch(var)

{

case 1:

insert();

break;

case 2:

{

printf("\nEnter the name you want to delete from the phonebook : ");

char del\_name[50];

scanf("%s",del\_name);

delete\_data(del\_name);

break;

}

case 3:

display();

break;

case 4:

display\_rev();

break;

case 5:

update();

break;

case 6:

search();

break;

case 7:

search\_by\_char();

break;

case 8:

sortList();

break;

case 9:

printf("\nTHANK YOU!! LOOKING FORWARD TO HELP YOU!!\n");

exit(0);

default:

printf("\nINVALID OPTION!! TRY AGAIN.\n\n");

break;

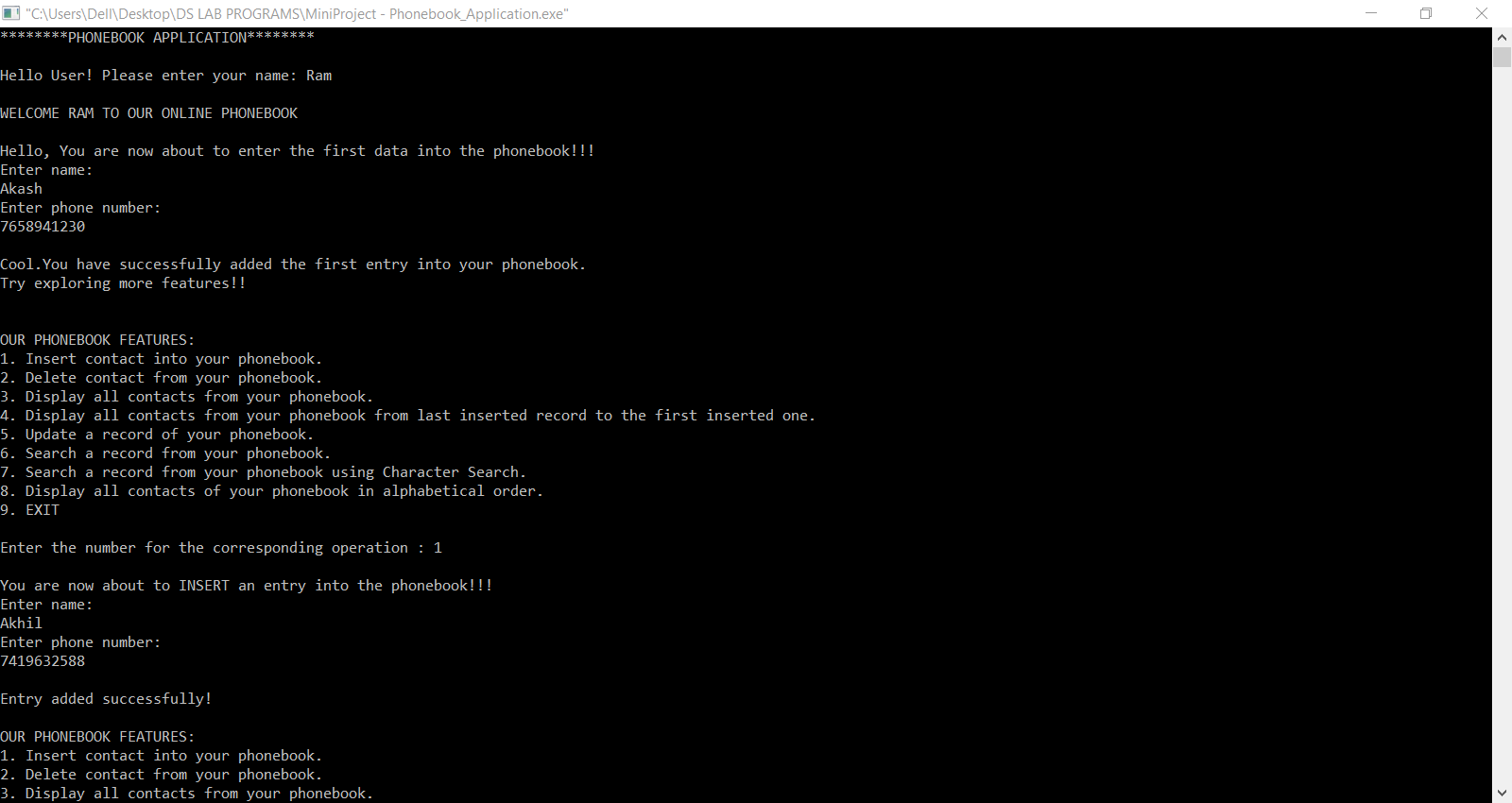
}

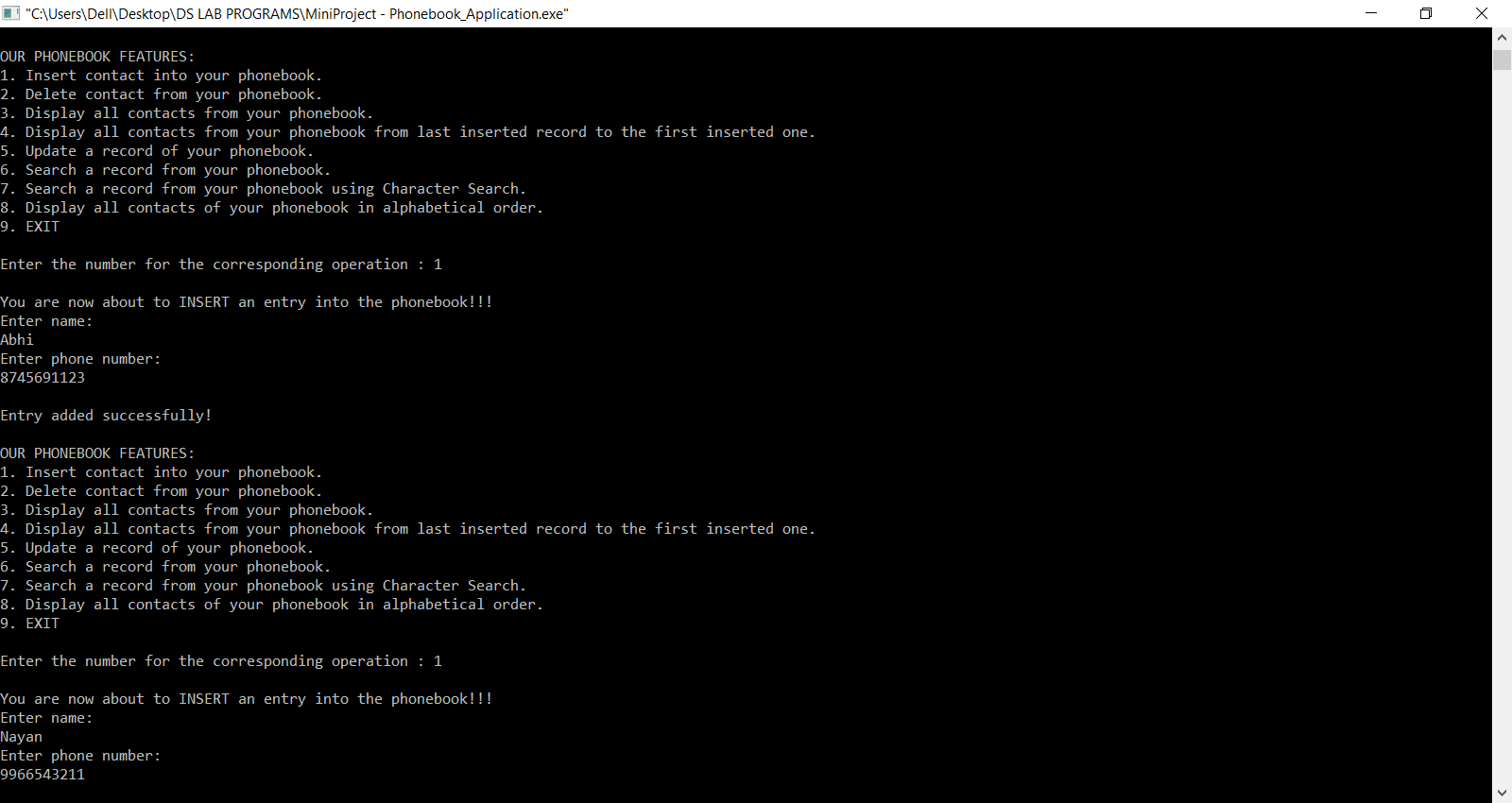
}while(var!=9);

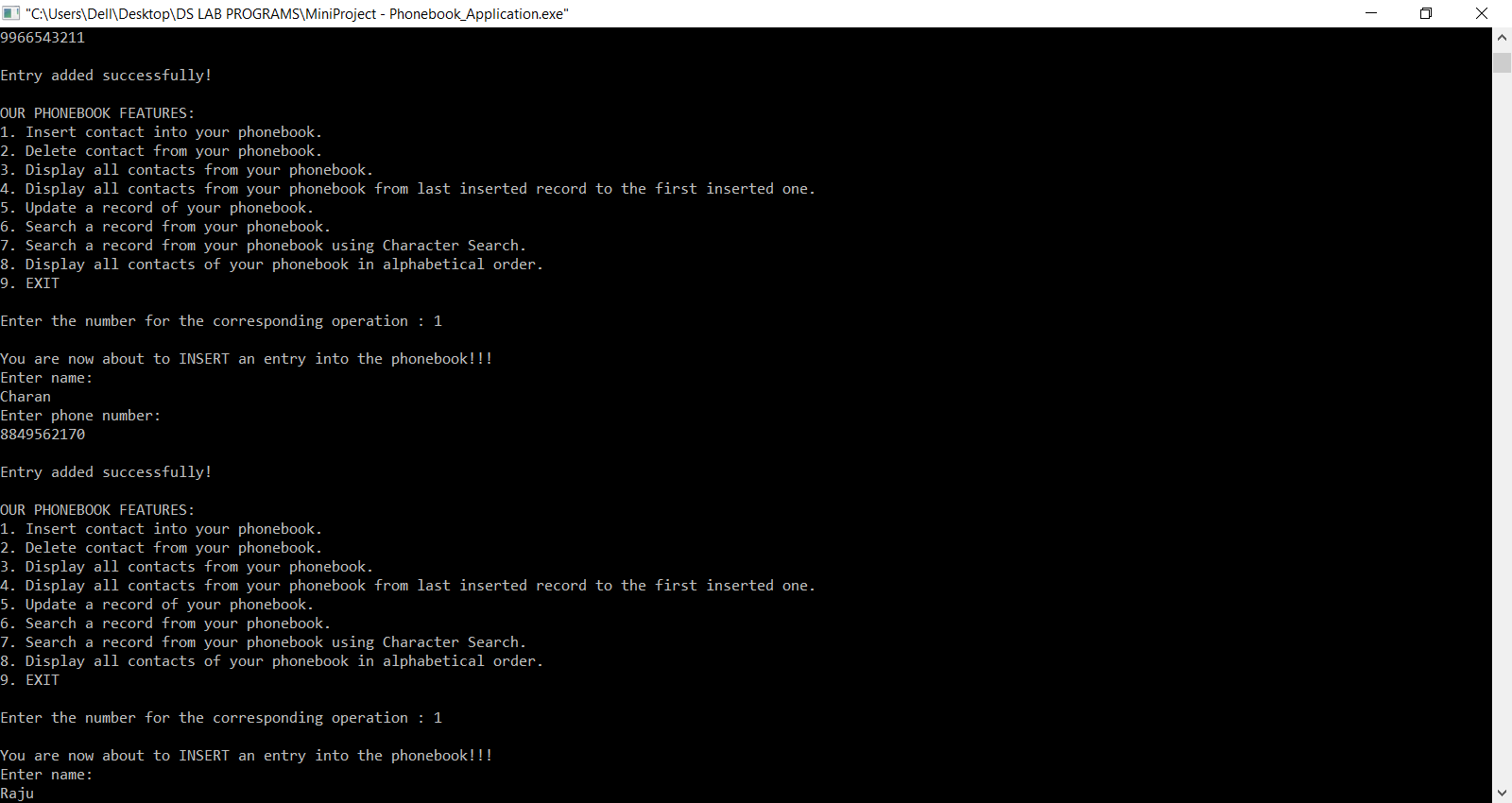
return 0;

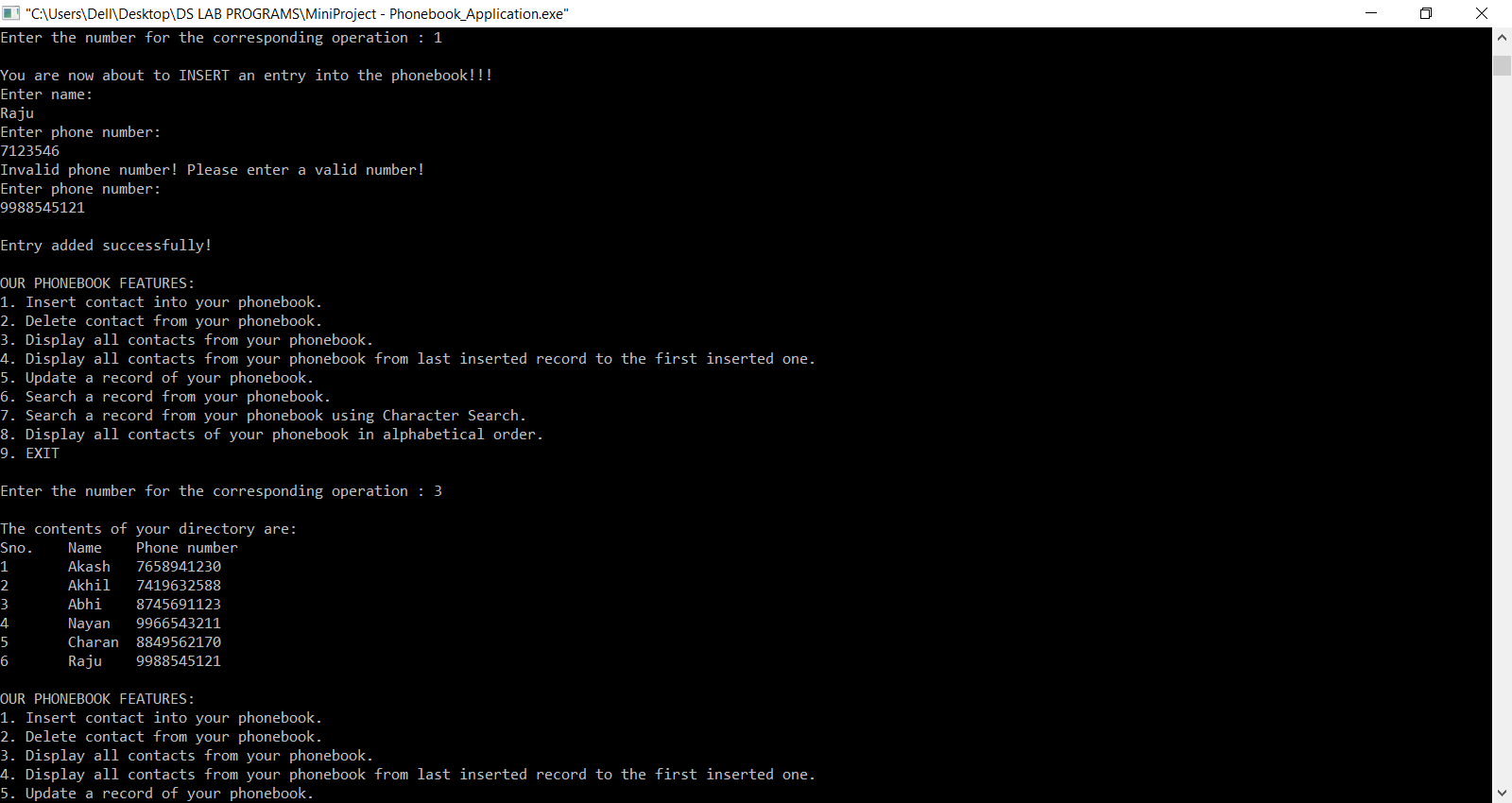
}

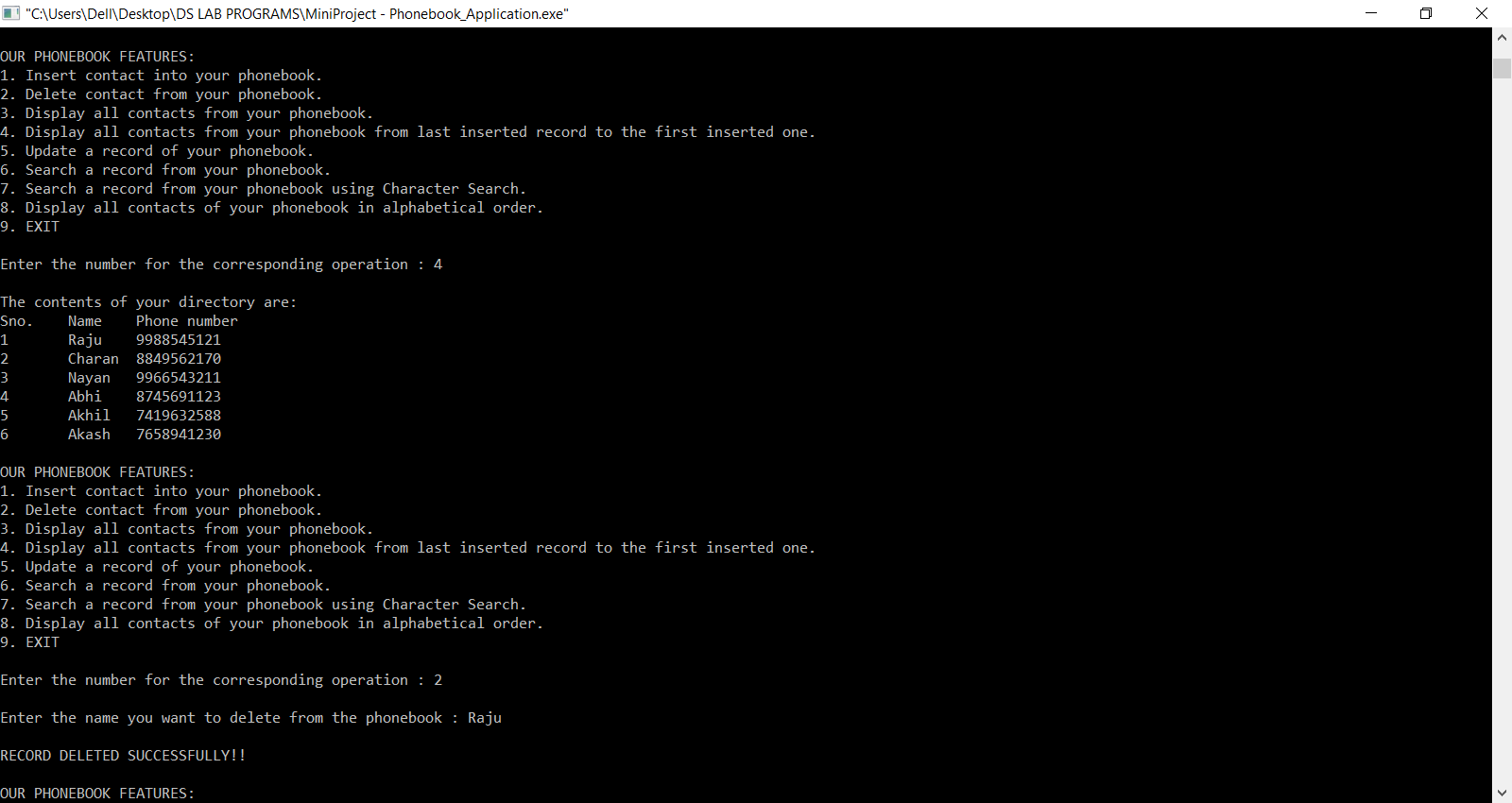
**OUTPUT OF THE PROGRAM**

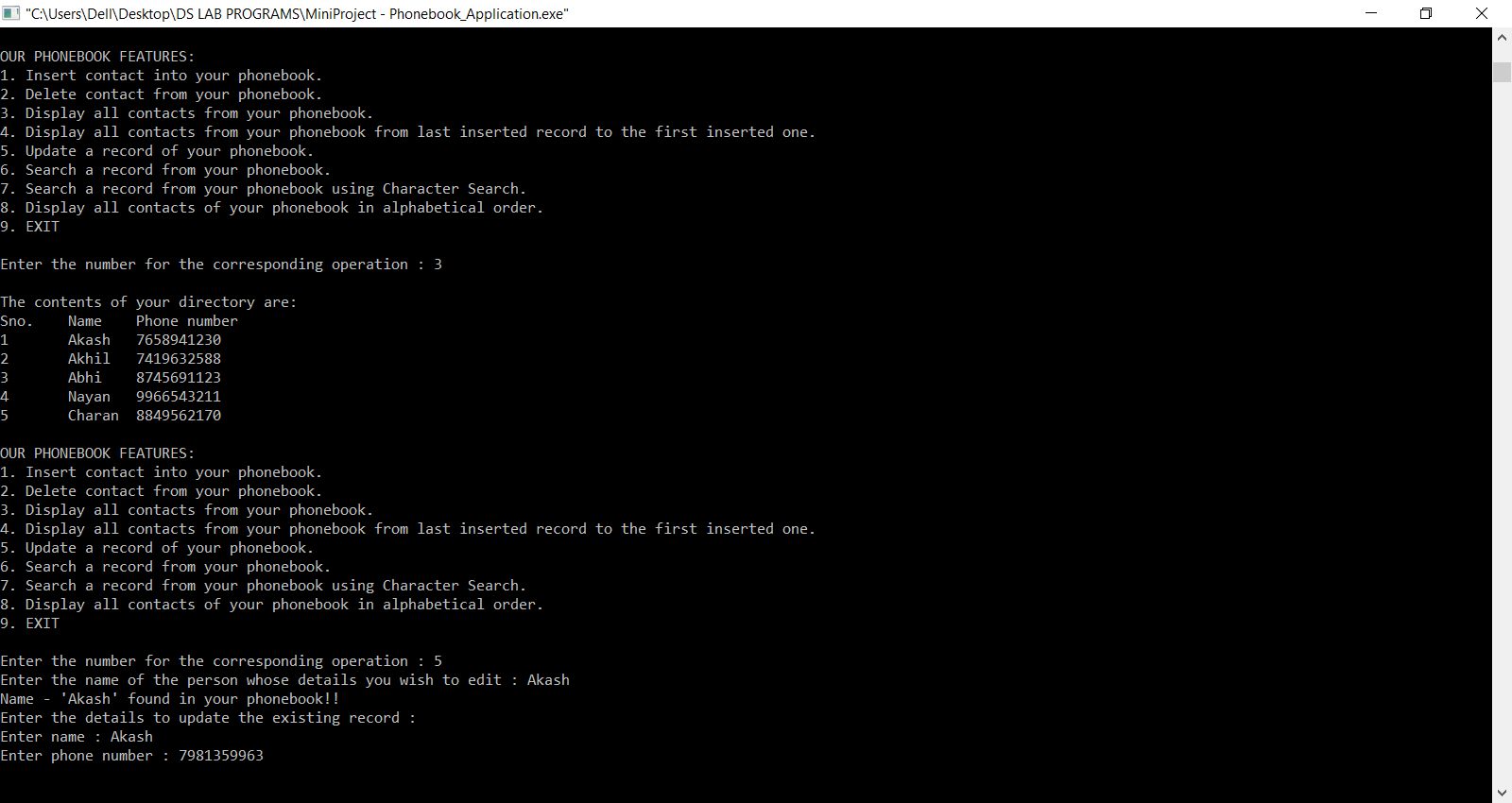


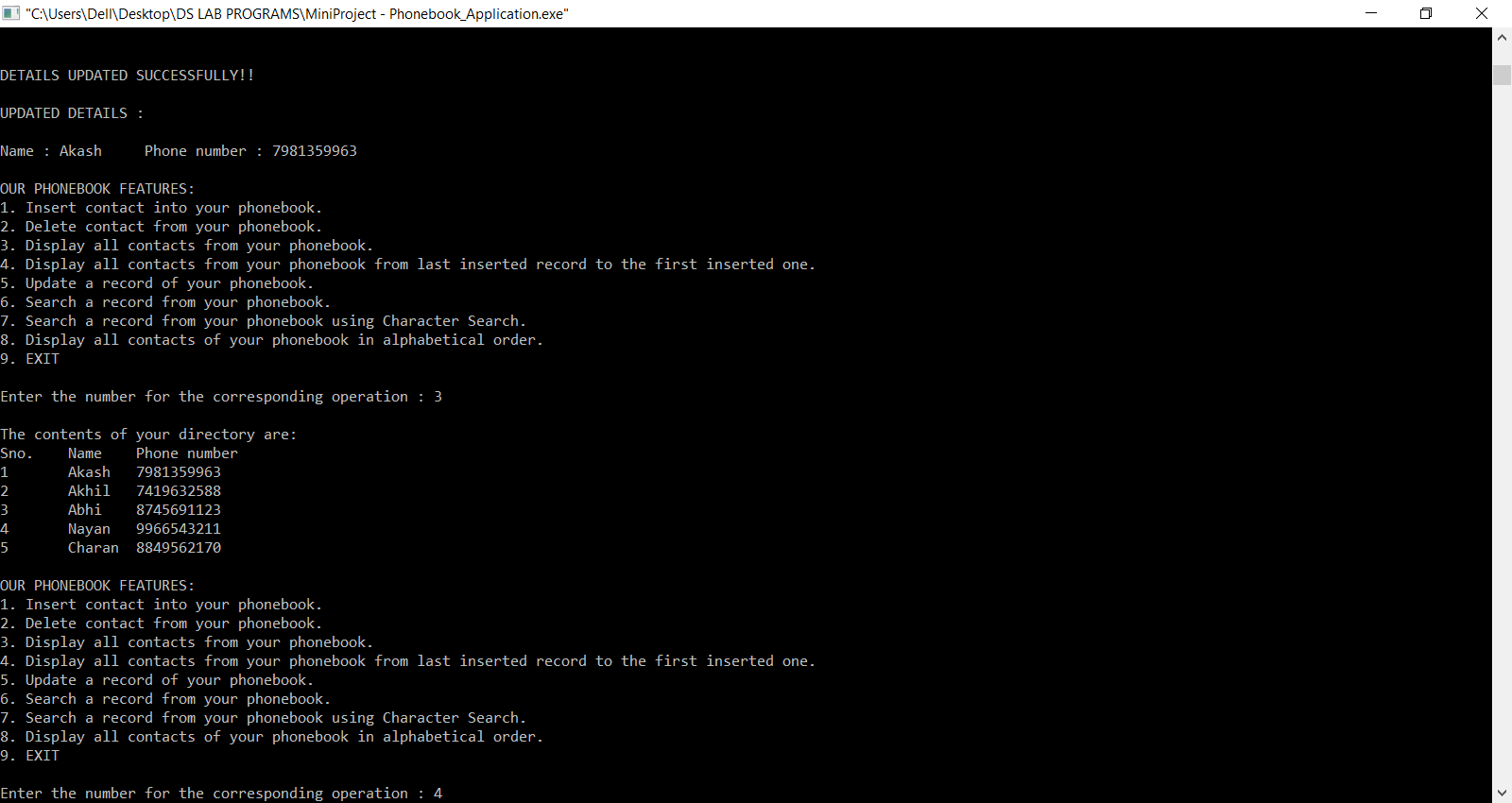


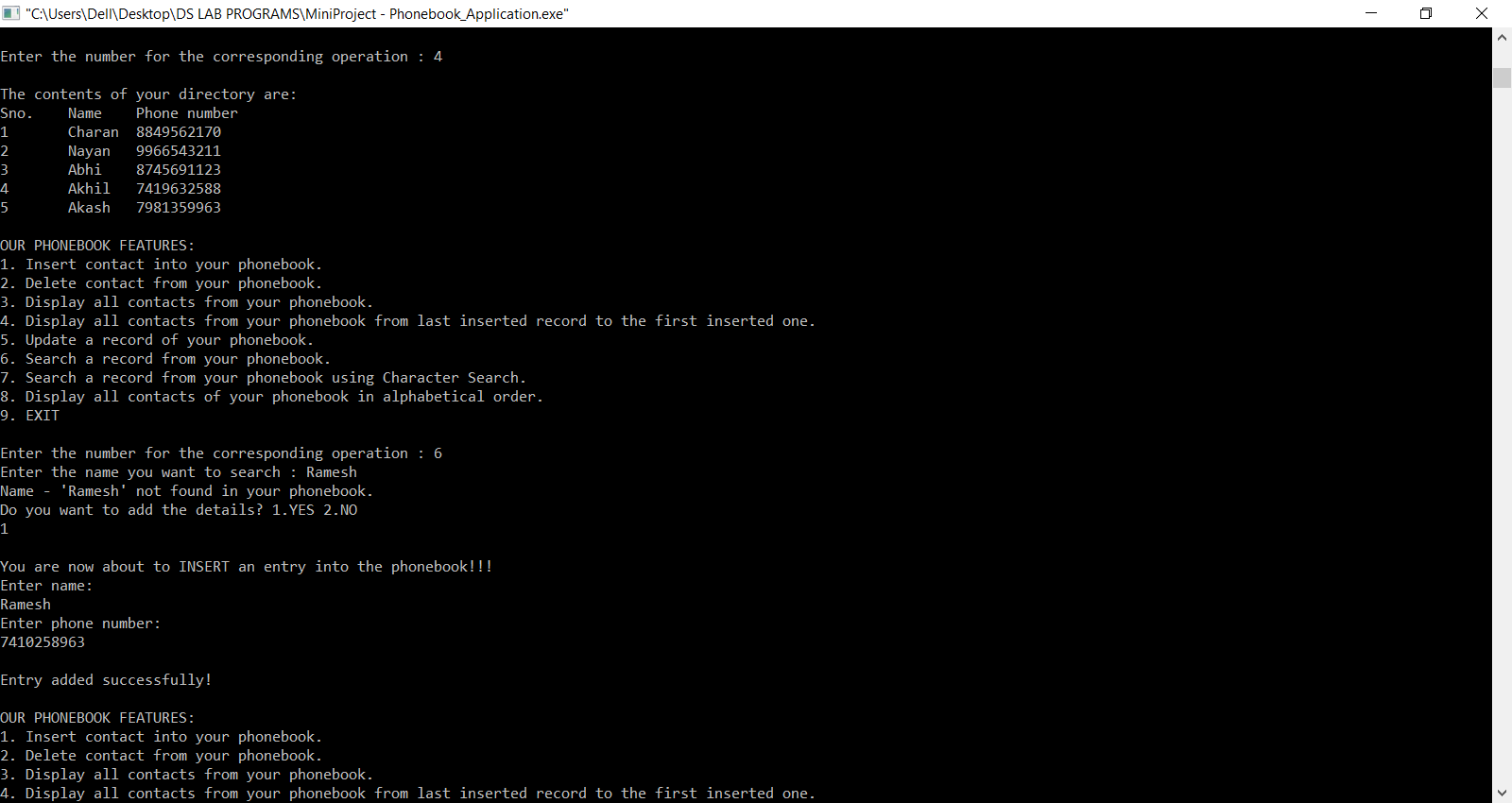


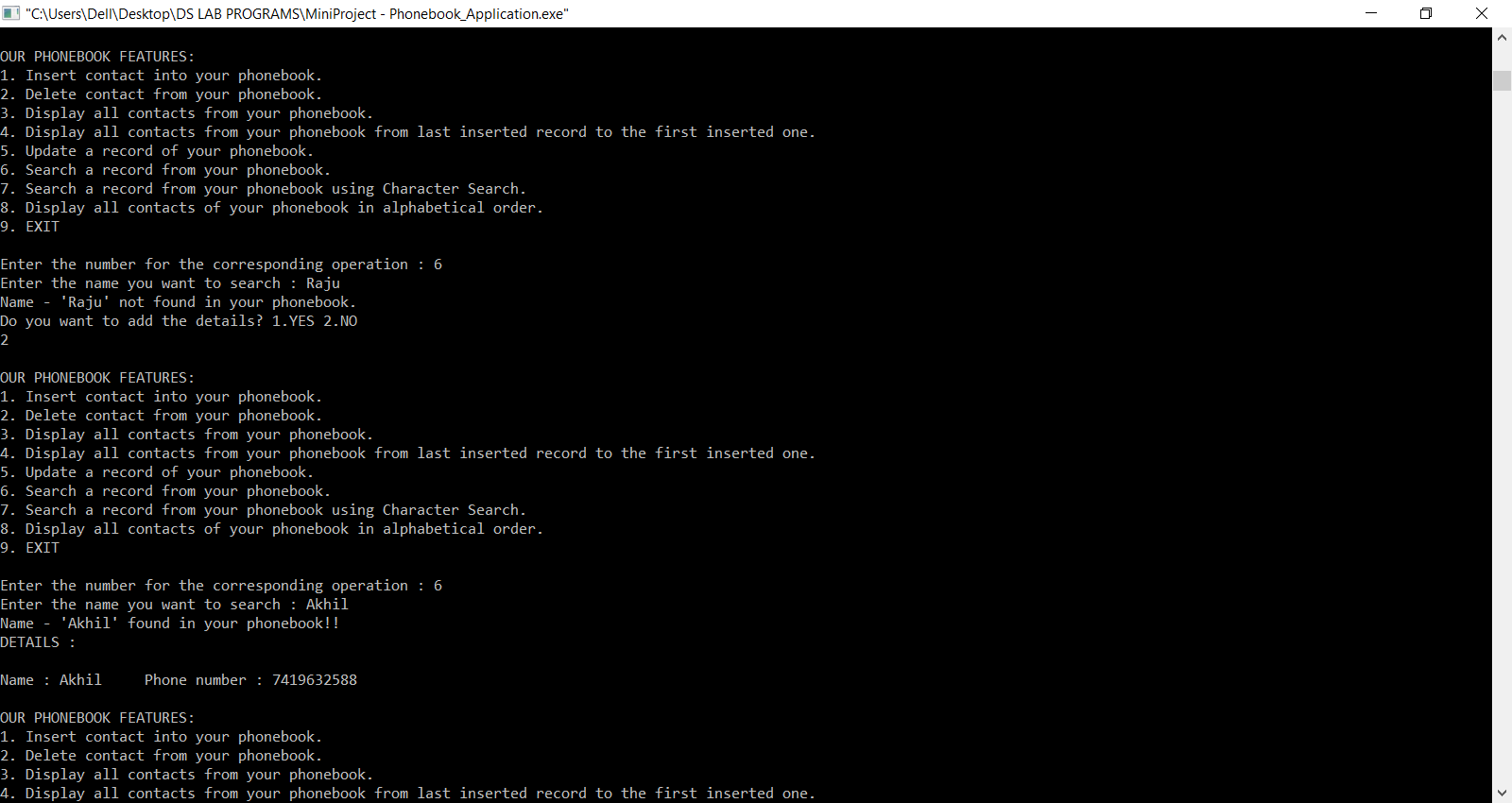


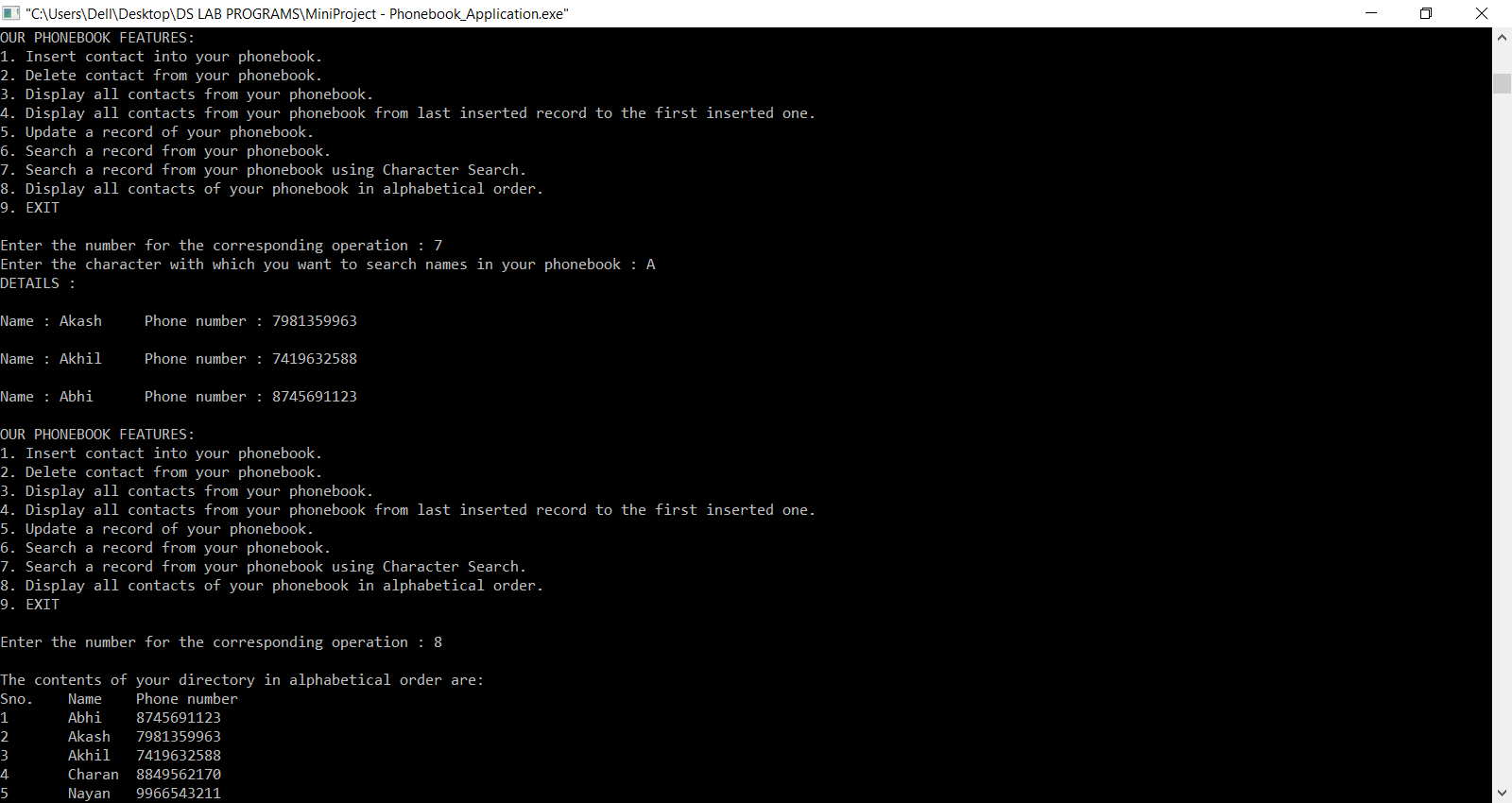


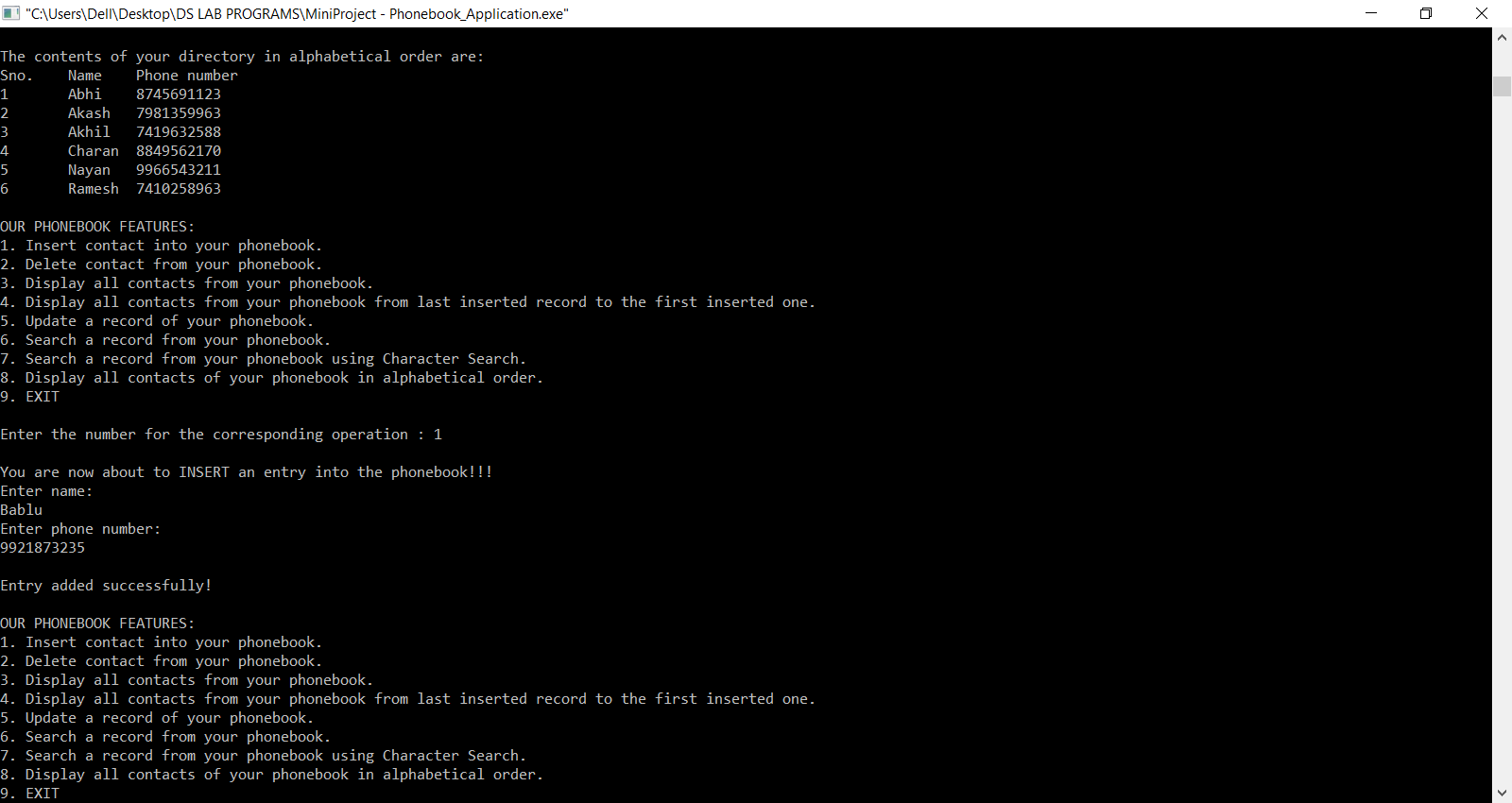


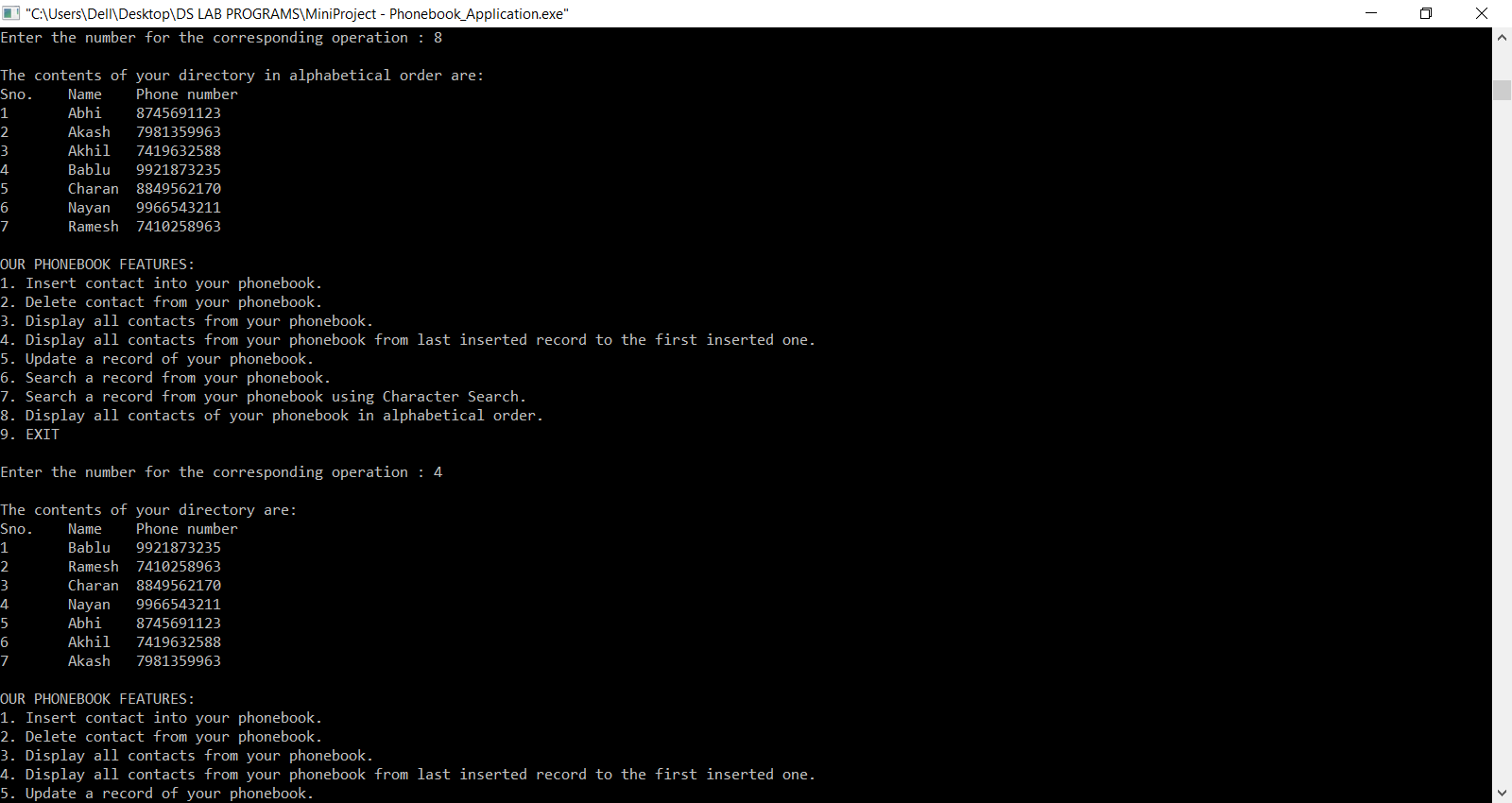


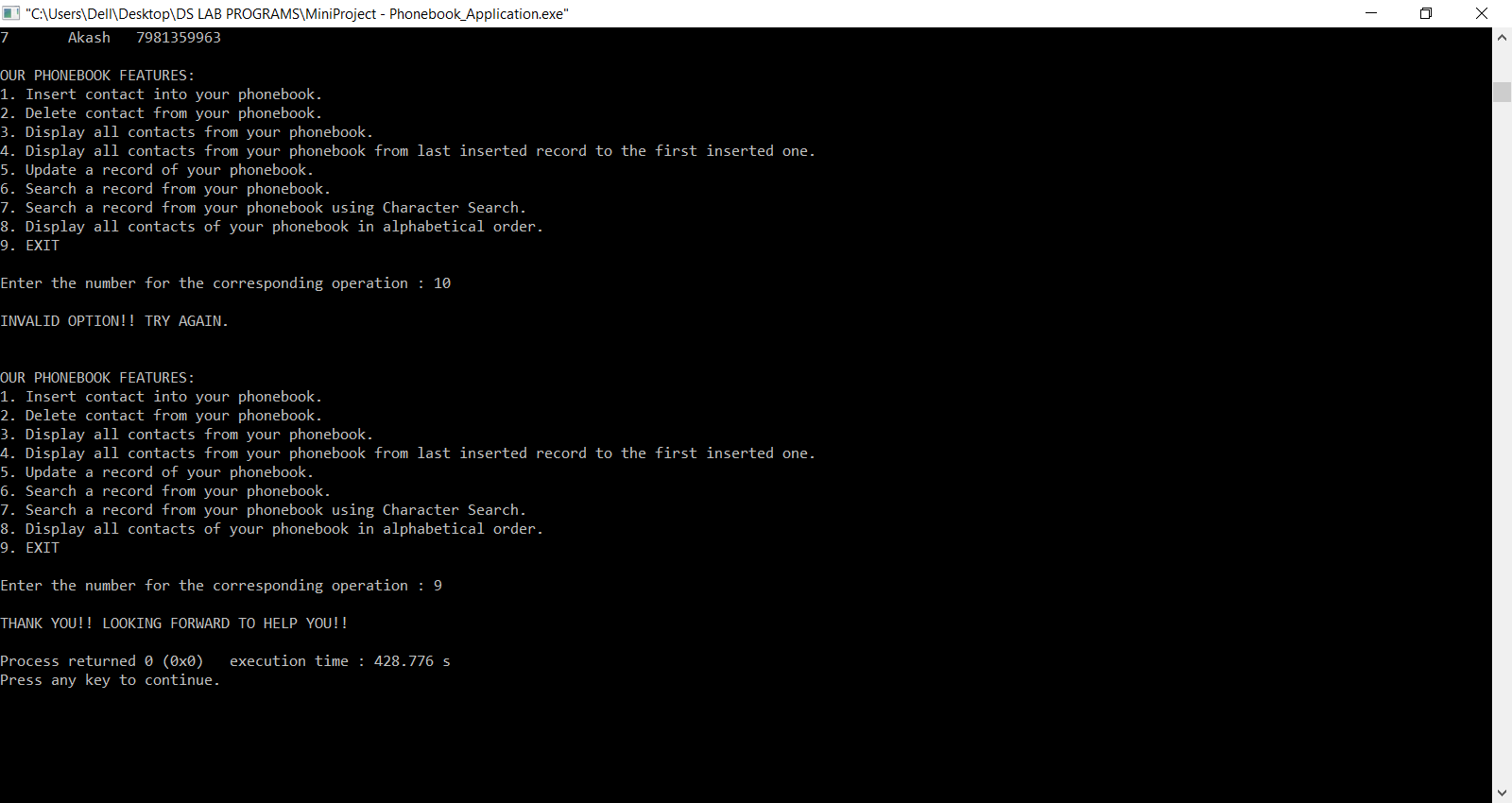












**FUTURE WORK**

We would like to extend this project by adding real time data storage applications using an online Database service so that the user could store his contacts permanently.

**REFERENCES**

* Fundamentals of DATA STRUCTURES in C (Second Edition) by Ellis Horowitz
* <https://www.geeksforgeeks.com>
* <https://www.slideshare.net/>
* <https://stackoverflow.com/>