A Course Based Project Report on

PHONE DIRECTORY MANAGEMENT SYSTEM

Submitted to the

Department of Information Technology

in partial fulfilment of the requirements for the completion of course
DATA STRUCTURES LABORATORY(22ES2CS102)

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

B.HARSHINI	22071A1271
CH.DEEPTHI	22071A1272
CH.JYOTHIKA	22071A1273
C.ATHARVA REDDY	22071A1274
D.VENKAT	22071A1275

Under the guidance of

Mrs. M.SUSMITHA

Assistant Professor, Department of IT, VNRVJIET

TAMASOMA JYOTIRGAMAYA
Estd.1995

DEPARTMENT OF INFORMATION TECHNOLOGY

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

AUGUST 2023

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses, Approved by AICTE, New Delhi, Affiliated to JNTUH, Recognized as "College with Potential for Excellence" by UGC, ISO 9001:2015 Certified, QS I GUAGE Diamond Rated
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

DEPARTMENT OF INFORMATION TECHNOLOGY

TAMASOMA JYOTIRGAMAYA
Estd.1995

CERTIFICATE

This is to certify that the project report entitled "**PHONE DIRECTORY MANAGEMENT SYSTEM**" is a bonafide work done under our supervision and is being submitted by **B.Harshini(22071A1271), CH.Deepthi(22071A1272), CH.Jyothika(22071A1273),C.AtharvaReddy(22071A1274),D.Venkat(22071A1275)** in partial fulfilment for the award of the degree of **Bachelor of Technology** in Information Technology, of the VNRVJIET, Hyderabad during the academic year 2022-2023.

Mrs.M.Susmitha	**Dr.D Srinvasa Rao**
Assistant Professor	Associate Professor & HOD
Department of IT	Department of IT

External Reviewer

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with ‘A++’ Grade,

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

DEPARTMENT OF INFORMATION TECHNOLOGY

TAMASOMA JYOTIRGAMAYA

Estd. 1995

DECLARATION

We declare that the course based project work entitled “**PHONE DIRECTORY MANAGEMENT SYSTEM**” submitted in the Department of Information Technology, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in Information Technology** is a bonafide record of our own work carried out under the supervision of **Mrs.M.Susmitha, Assistant Professor, Department of IT, VNRVJIET.** Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad.

Harshini	[illegible]	[illegible]	[illegible]	[illegible]
B.HARSHINI	**CH.DEEPTHI**	**CH.JYOTHIKA**	**C.ATHARVA REDDY**	**D.VENKAT**
(22071A1271)	(22071A1272)	(22071A1273)	(22071A1274)	(22071A1275)

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our beloved President, Sri. D. Suresh Babu, VNR Vignana Jyothi Institute of Engineering & Technology for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we record our deep sense of gratitude to our beloved Principal, Dr. C.D Naidu, for permitting us to carry out this project.

We express our deep sense of gratitude to our beloved Professor Dr. SRINIVASA RAO DAMMAVALAM, Associate Professor and Head, Department of Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad-500090 for the valuable guidance and suggestions, keen interest and through encouragement extended throughout the period of project work.

We take immense pleasure to express our deep sense of gratitude to our beloved Guide, **Mrs.M.Susmitha**, Assistant Professor in Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad, for his/her valuable suggestions and rare insights, for constant source of encouragement and inspiration throughout my project work.

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

B. Harshini	(22071A1271)
CH. Deepthi	(22071A1272)
CH. Jyothika	(22071A1273)
C. Atharva Reddy	(22071A1274)
D.Venkat	(22071A1275)

ABSTRACT

Phone Directory is a C Data Structures based project to store our contacts. We can use it to replace our hard phonebook or even use it as an office-wide phone directory. This will help user to easily search and manage contacts details of an individual. The phone numbers are present in formats like alphabetical order, numerical order. So that user can easily find the required person either by entering first letter of the name or the total name. Or Either by entering total number or entering first digit of the number. In order to keep updated the phone book directory, the admin will have the authority to add and delete the phone numbers within the phone directory. To make all operations as easier as possible, user friendly approach has been taken into account by which users have to select the option given in the menu to make their operations successful. For searching operation, users will able to get any particular record using the name or phone number but the only condition is that, customers record must be available within the file system. If no such record is available, proper error message will be displayed as "Contact not found".

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION

1.1 PURPOSE

The "Phone Directory Management System" is a command-line interface project implemented in the C programming language, focusing on data structures to efficiently manage a phone directory. The project aims to provide users with a convenient and user-friendly tool to store, retrieve, update, and delete contact information.

1.2 SCOPE

The "Phone Directory Management System" is a command-line interface project implemented in the C programming language, focusing on data structures to efficiently manage a phone directory. The project aims to provide users with a convenient and user-friendly tool to store, retrieve, update, and delete contact information.

The system uses data structures like linked lists or hash tables to organize the phone directory efficiently, ensuring quick access to contact details while minimizing memory usage. Users can perform various operations through the command-line interface, such as adding a new contact, searching for a contact by name, updating an existing contact's information, and deleting contacts.

While the core features are well-defined, there is potential for scope expansion through additional features like contact categorization, sorting options, import/export functionality, or integration with external services. The project's scope allows for customization and adaptation based on user needs, making it a versatile tool for contact management.

1.3 DEFINITION:

The "Phone Directory Management System" is a C-based command-line project designed to efficiently organize and manage contact information. It employs data structures like linked lists or hash tables to optimize directory operations for quick access and minimal memory use. Users interact through a command-line interface to perform tasks such as adding, searching, updating, and deleting contacts. The project is expandable, accommodating features like contact categorization, sorting, and import/export capabilities. Its adaptable nature allows for customization, making it versatile for various contact management needs, offering a convenient toolset for users to store, retrieve, and manage contact details effectively.

1.4 TECHNOLOGIES TO BE USED:

- C programming language

1.5 DEVELOPMENT TOOLS:

1. TURBO C.

1.6 OVERVIEW:

A Phone Directory Management System presents a comprehensive solution for efficiently organizing and managing contact information. Leveraging optimized data structures like linked lists or hash tables, the system ensures swift access to contacts while conserving memory resources. Its user-friendly command-line interface simplifies tasks such as adding, searching, updating, and deleting contacts, making it accessible to users of varying technical backgrounds. The system's customizable features, including contact categorization, sorting options, and import/export functionalities, allow for tailored use based on specific needs. The "Phone Directory Management System" is a robust command-line application developed in C, tailored for efficient organization and management of contact information. Using optimized data structures such as linked lists or hash tables, the system ensures quick access to contacts while conserving memory resources.

CHAPTER 2 – OVERALL DESCRIPTION

The Phone Directory Management System is a C-based command-line application designed for efficient contact management. It utilizes linked lists and hash tables to organize and access contact information swiftly while optimizing memory usage. Users interact through a user-friendly command-line interface to perform tasks such as adding, searching, updating, and deleting contacts. The system's core functionalities prioritize simplicity and effectiveness, catering to users seeking a straightforward tool for contact management. Future enhancements, such as contact categorization and integration with external services, are possible within the system's adaptable framework, providing a versatile solution for various contact management needs.

2.1 PRODUCT PERSPECTIVE:

The Phone Directory Management System is a command-line application designed to efficiently manage contact information using C programming language. It operates as a standalone system, interacting directly with users through a command-line interface. The system organizes contact data using linked lists and hash tables, optimizing memory usage and ensuring quick access to contacts. It provides essential functionalities like adding, searching, updating, and deleting contacts, enhancing user productivity in contact management tasks.

2.2 SOFTWARE INTERFACE:

The system is implemented in C and uses standard input/output functions for user interaction. It leverages data structures such as linked lists and hash tables to handle contact storage and retrieval efficiently. The software integrates error handling mechanisms to ensure robust performance and data integrity.

2.3 HARDWARE INTERFACE:

The Phone Directory Management System operates on a standard computer system compatible with the C programming environment. It requires basic hardware specifications suitable for running C programs, including a compatible compiler (like Turbo C).

2.4 SYSTEM FUNCTIONS:

- **Create Contact**: Allows users to add new contacts to the directory.
- **Search Contacts**: Supports various search methods based on name or phone number.
- **Delete Contacts**: Enables users to remove contacts from the directory.
- **Display Contacts**: Provides options to display contacts sorted by name or phone number.
- **Customization**: Allows for potential customization, such as categorization or import/export functionalities, to extend system capabilities.

2.5 USER CHARACTERISTICS:

The system targets users seeking a straightforward and efficient tool for managing contact information. Users should have basic familiarity with command-line interfaces and contact management concepts. The system's design prioritizes ease of use and accessibility, making it suitable for a wide range of technical skill levels.

2.6 CONSTRAINTS:

The system's functionality is limited to command-line interaction and basic contact management tasks. Advanced features like contact categorization or integration with external services would require additional development effort.

2.7 ASSUMPTIONS AND DEPENDENCIES:

It relies on the availability of standard input/output functionalities for user interaction. The system operates independently without external dependencies, focusing solely on managing contact information within the scope of the command-line interface. Future enhancements or integrations would depend on the expansion of the existing codebase and compatibility with related systems or services.

CHAPTER 3 – NON FUNCTIONAL REQUIREMENTS

3.1 Performance Requirements

1. **Efficiency**: The system should efficiently handle operations like adding, searching, updating, and deleting contacts, even with large datasets.
2. **Responsiveness**: The user interface should respond promptly to user inputs, ensuring a smooth and seamless experience.
3. **Memory Usage**: Optimized data structures should minimize memory usage, enhancing overall system performance.

3.2 Safety Requirements

1. **Data Integrity**: The system must ensure the integrity and accuracy of stored contact information to prevent data loss or corruption.
2. **Error Handling**: Effective error handling mechanisms should be in place to manage unexpected user inputs or system errors without crashing.

3.3 Security Requirements

1. **Data Privacy**: Contact information should be stored securely to protect user privacy and confidentiality.
2. **Access Control**: Implement proper access controls to restrict unauthorized access to sensitive contact details.

3.4 Software Quality Attributes

1. **Reliability**: The system should consistently perform contact management tasks accurately without unexpected failures.
2. **Maintainability**: Codebase should be well-structured and documented, allowing for easy maintenance and future enhancements.
3. **Usability**: The user interface should be intuitive and user-friendly, requiring minimal training for users to operate effectively.

Business Rules

1. **Unique Contacts**: Each contact's name and phone number combination should be unique within the directory.
2. **Data Consistency**: Contact details should be consistent and up-to-date, reflecting any changes made through the system's operations.
3. **Compliance**: The system should comply with relevant data protection regulations and business policies regarding contact information handling.

CHAPTER 4 -CODE AND OUTPUT

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
typedef struct phone
{
   char a[20],b[20];
   struct phone *link;
}node;
node *book[26]={NULL},*phone[10]={NULL};
void search();
void  search_by_letter();
void search_by_name();
void search_by_first_digit();
void search_by_number();
void del();
void delete_by_name(char [],int);
void delete_by_number(char [],int);
void create();
void display();
void display_by_names();
void display_by_number();
void numpage(char[],char[]);
void search()
{
   int choice;
   printf("Menu is :\n1.search by first letter of the name\n2.search by full name\n3-
Search by first digit of number\n4-search by full number\npress '-1' to exit\n”)
```

```
scanf("%d",&choice);
switch (choice)
          {
          case 1:search_by_letter();
            break;
          case 2:search_by_name();
          break;
          case 3:search_by_first_digit();
          break;
          case 4:search_by_number();
          break;
          case -1:exit(0);
          default:printf("Invalid Input\n");
            break;
          }
        }

        void search_by_letter()
        {
          node *temp;
          char  ch[20];
          int n;
          printf("Enter a letter:\n");
          scanf("%s",ch);
          n=ch[0]-'a';
          temp=book[n];
          if(book[n]==NULL)
             printf("Empty\n");
          else
          {
            temp=book[n];
            printf("Available Contacts starting with given letter is \n");
        while(temp!=NULL)
        {
            printf("%s\t%s\n",temp->a,temp->b);
            temp=temp->link;
        }
    }
}

void search_by_name()
{
    char str[20];
    int n;
    node *temp;
    printf("Enter contact:\n");
    scanf("%s",str);
    n=str[0] - 'a';
    temp=book[n];
    while(temp!=NULL )
    {
        if(strcmp(str,temp->a)==0)
        {
            printf("Contact found\n%s\t%s\n",temp->a,temp->b);
            break;
        }
        temp=temp->link;
    }
    if(temp==NULL)
        printf("Contact not found.Sorry!...\n");
}

void del()
{
    int choice;
    int  n=2;
    char str[20];
    printf("Menu is \n1.delete by contact name\n2-Delete by number\n3-exit\n");
    printf("Enter choice:\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
        printf("Enter the contact name to be deleted:\n");
        scanf("%s",str);
        delete_by_name(str,n);
        break;
        case 2:
        printf("Enter the contact number to be deleted:\n");
        scanf("%s",str);
        delete_by_number(str,n);
        break;
        case 3:exit(0);
        default:
        printf("Invalid Input\n");
    }
}

void delete_by_name(char str[],int k)
{
    int n;
    node *temp,*prev,*cur;
    n=str[0] - 'a';
    cur=book[n];
    if(cur==NULL)
         printf("Nothing is there to be deleted\n");
```

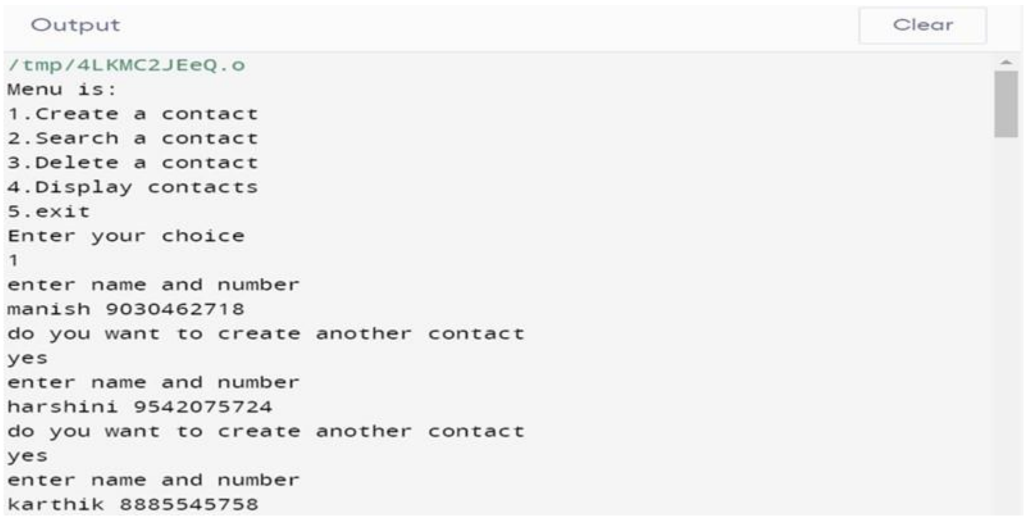
```
else if(strcmp(cur->a,str)==0)
{
   book[n]=cur->link;
   if(k==2)
   printf("%s contact is deleted\n",cur->a);
   if(k==2)
   delete_by_number(cur->b,--k);
   free(cur);
}
else
{
   while(cur->link!=NULL && strcmp(cur->a,str)!=0)
   {
      prev=cur;
      cur=cur->link;
   }
   if(cur->link!=NULL && strcmp(cur->a,str)==0)
   {
      prev->link=cur->link;
      if(k==2)
      printf("%s contact is deleted\n",cur->b);
      if(k==2)
      delete_by_number(cur->b,--k);
      free(cur);
   }
   else if(cur->link==NULL && strcmp(cur->a,str)==0)
   {
      if(k==2)
      printf("%s contact is deleted\n",cur->a);
      prev->link=NULL;
      if(k==2)
      delete_by_number(cur->b,--k);
```

```
          free(cur);
        }
        else
        {
          printf("Contact not found\n");
        }
    }
}

void delete_by_number(char str[],int k)
{
    int n;
    node *temp,*prev,*cur;
    n=str[0] - '0';
    cur=phone[n];
    if(cur==NULL)
         printf("No such type contact\n");
    else if(strcmp(cur->b,str)==0)
    {
        phone[n]=cur->link;
        if(k==2)
        printf("%s contact is deleted\n",cur->b);
        if(k==2)
        delete_by_name(cur->a,--k);
        free(cur);
    }
    else
    {
        while(cur->link!=NULL && strcmp(cur->b,str)!=0)
        {
            prev=cur;
            cur=cur->link;
        }
        if(cur->link!=NULL && strcmp(cur->b,str)==0)
        {
          prev->link=cur->link;
          if(k==2)
          printf("%s contact is deleted\n",cur->b);
          if(k==2)
          delete_by_name(cur->a,--k);
          free(cur);
        }
        else if(cur->link==NULL && strcmp(cur->b,str)==0)
        {
          if(k==2)
          printf("%s contact is deleted\n",cur->b);
          prev->link=NULL;
          if(k==2)
          delete_by_name(cur->a,--k);
          free(cur);
        }
        else
        {
          printf("Contact not found\n");
        }
    }
}

void display()
{
    int n;
    printf("menu:\n1-display according to numbers\n2-display according to names\n3-
exit\n");
    printf("Enter your choice\n");
    scanf("%d",&n);
    switch(n)
    {
        case 1:display_by_number();break;
        case 2:display_by_names();break;
        case 3:exit(0);
        default:printf("Invalid Input\n");
        break;
    }
}

void display_by_names()
{
    node *temp;
    int i,n=0;
    for(i=0;i<26;i++)
    {
        if(book[i]==NULL)
        continue;
        else{
            temp=book[i];
            printf("%c\n",temp->a[0]);
            while(temp!=NULL)
            {
                n=0;
                while(temp->a[n]!='\0')
                {
                    printf("%c",temp->a[n]);
                    n++;
                }
                printf("->");
                for(n=0;temp->b[n]!='\0';n++)
                printf("%c",temp->b[n]);
                printf("\n");
                temp=temp->link;
            }
        }
    }
}

void display_by_number()
{
    node *temp;
    int i,n=0;
    for(i=0;i<10;i++)
    {
        if(phone[i]==NULL)
        continue;
        else{
            temp=phone[i];
            printf("%c\n",temp->b[0]);
            while(temp!=NULL)
            {
                n=0;
                while(temp->b[n]!='\0')
                {
                    printf("%c",temp->b[n]);
                    n++;
                }
                printf("->");
                for(n=0;temp->a[n]!='\0';n++)
                printf("%c",temp->a[n]);
                printf("\n");
                temp=temp->link;
            }
        }
    }
}

void create()
{
    char num[20],name[20],y[20];
    int n,i;
    node *newnode,*cur,*pre;
    do{
        printf("enter name and number\n");
        scanf("%s%s",name,num);
        newnode=(node*)malloc(sizeof(node));
        for(i=0;name[i]!='\0';i++)
        newnode->a[i]=name[i];
        newnode->a[i]='\0';
        for(i=0;num[i]!='\0';i++)
        newnode->b[i]=num[i];
        newnode->b[i]='\0';
        newnode->link=NULL;
        numpage(name,num);
        n=name[0]-'a';
        if(book[n]==NULL)
        book[n]=newnode;
        else
        {
            cur=book[n];
            pre=cur;
            if(strcmp(cur->a,name)>0)
            {
                book[n]=newnode;
                newnode->link=cur;
            }
            else
            {
            while(cur->link!=NULL && strcmp(cur->a,name)<0)
            {
                pre=cur;
                cur=cur->link;
            }
            if(strcmp(cur->a,name)>0)
            {
                newnode->link=pre->link;
                pre->link=newnode;
            }
            else if(strcmp(cur->a,name)==0)
            printf("duplicate file\n");
            else if(cur->link==NULL && strcmp(cur->a,name)<0)
            cur->link=newnode;
            }
        }
    printf("do you want to create another contact\n");
    scanf("%s",y);
    }while(strcmp(y,"yes")==0);
}

int main()
{
    int choice;
    do
    {
    printf("Menu is:\n1.Create a contact\n2.Search a contact\n3.Delete a
contact\n4.Display contacts\n5.exit\nEnter your choice\n");
```

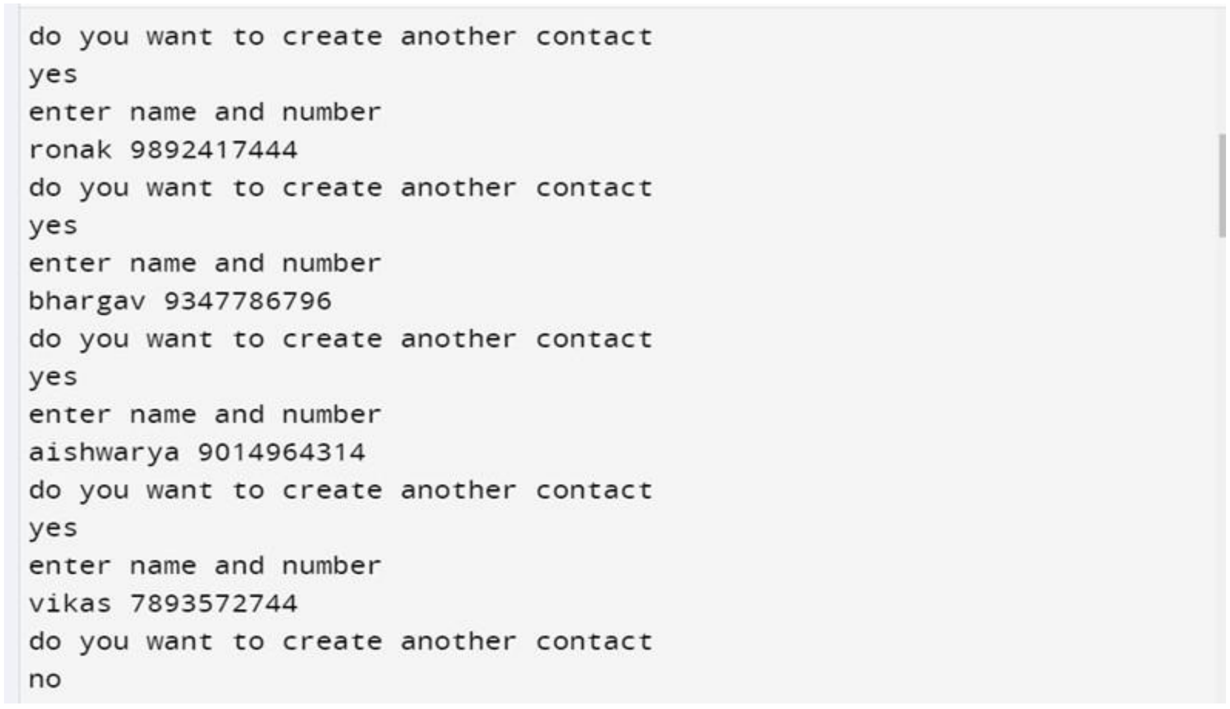
```
    scanf("%d",&choice);
    switch(choice)
    {
        case 1 : create(); break;
        case 2: search(); break;
        case 3:  del(); break;
        case 4: display(); break;
        case 5 : exit(0);
        default : printf("Enter valid choice:\n");
                  main();
    }
    }
    while(choice>0 && choice<5);
}

void numpage(char name[],char num[])
{
    node *cur,*pre,*newnode;
    int n,i=0;
    newnode=(node*)malloc(sizeof(node));
        for(i=0;name[i]!='\0';i++)
        newnode->a[i]=name[i];
        newnode->a[i]='\0';
        for(i=0;num[i]!='\0';i++)
        newnode->b[i]=num[i];
        newnode->b[i]='\0';
        newnode->link=NULL;
    n= newnode->b[0]-'0';
    if(phone[n]==NULL)
    phone[n]=newnode;
    else
    {
        cur=phone[n];
          pre=cur;
          if(strcmp(cur->b,newnode->b)>0)
          {
             phone[n]=newnode;
             newnode->link=cur;
          }
          else
          {
          while(cur->link!=NULL && strcmp(cur->b,newnode->b)<0)
          {
             pre=cur;
             cur=cur->link;
          }
          if(strcmp(cur->b,newnode->b)>0)
          {
             newnode->link=pre->link;
             pre->link=newnode;
          }
          else if(strcmp(cur->b,newnode->b)==0)
          printf("duplicate file\n");
          else if(cur->link==NULL && strcmp(cur->b,newnode->b)<0)
          cur->link=newnode;
   }
}
}
void search_by_number()
{
   char n[10];
   int i;
   node *temp;
   printf("Enter number to search\n");
```

```
    scanf("%s",n);
    i=n[0]-'0';
    temp=phone[i];
    while(temp!=NULL)
    {
        if(strcmp(temp->b,n)==0)
        {
            printf("Contact found\n");
            printf("%s\t%s\n",temp->b,temp->a);
            break;
        }
        temp=temp->link;
    }
    if(temp==NULL)
    printf("Contact not found\n");
}
void search_by_first_digit()
{
    int i;
    node *temp;
    char s[10];
    printf("Enter first digit of number\n");
    scanf("%s",s);
    i=s[0]-'0';
    temp=phone[i];
    printf("Contact numbers starting with %s are \n",s);
    while(temp!=NULL)
    {
        printf("%s\t%s\n",temp->b,temp->a);
        temp=temp->link;
    }
}
```

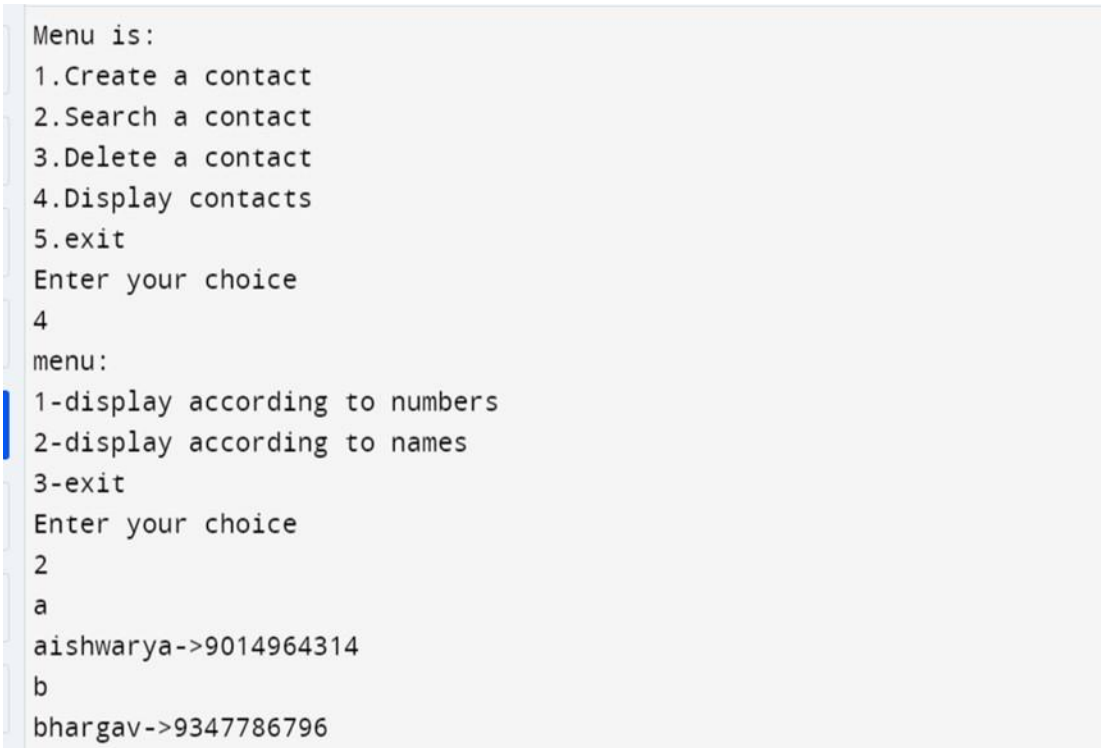
OUTPUT



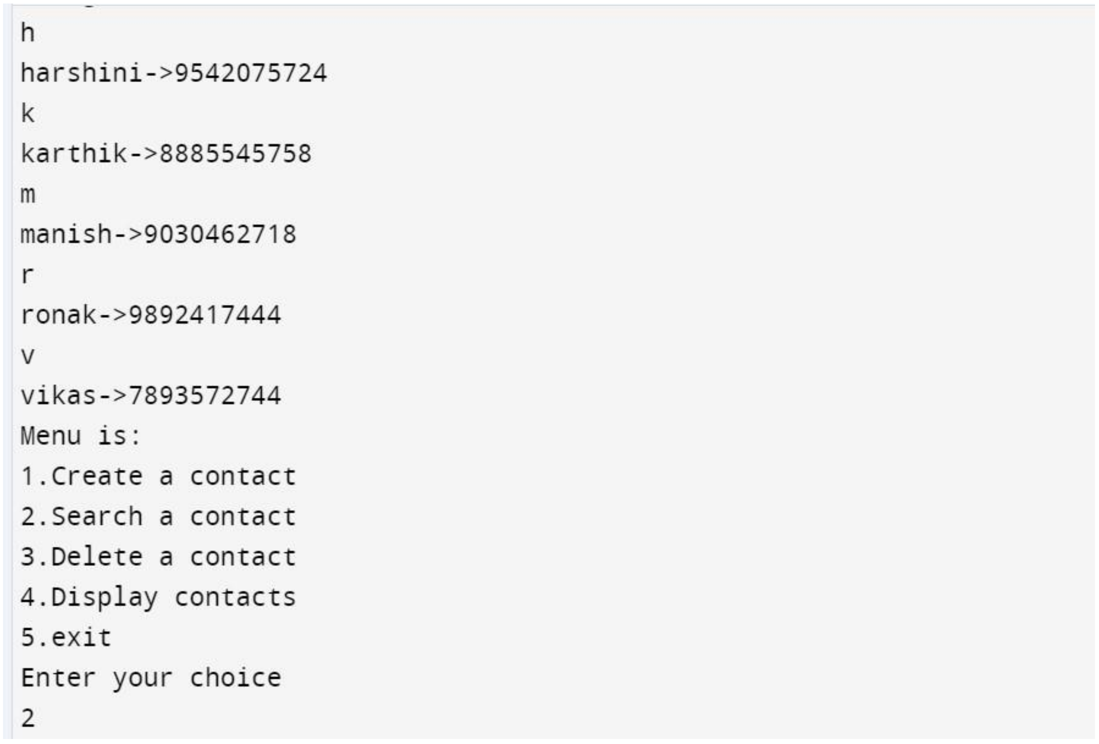
```
/tmp/4LKMC2JEeQ.o
Menu is:
1.Create a contact
2.Search a contact
3.Delete a contact
4.Display contacts
5.exit
Enter your choice
1
enter name and number
manish 9030462718
do you want to create another contact
yes
enter name and number
harshini 9542075724
do you want to create another contact
yes
enter name and number
karthik 8885545758
```



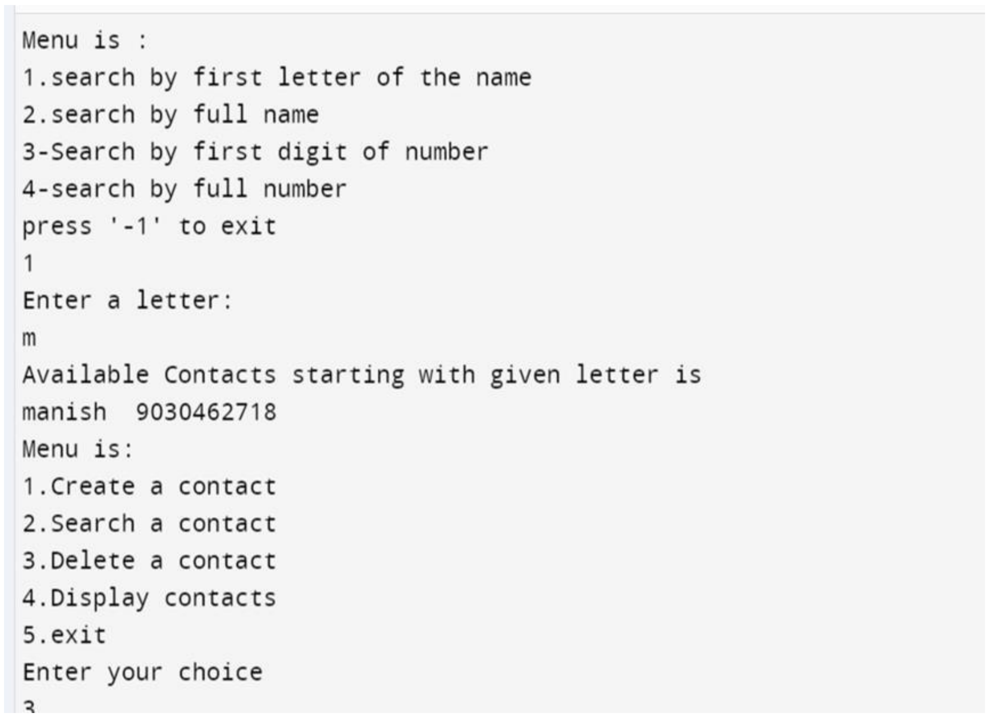
```
do you want to create another contact
yes
enter name and number
ronak 9892417444
do you want to create another contact
yes
enter name and number
bhargav 9347786796
do you want to create another contact
yes
enter name and number
aishwarya 9014964314
do you want to create another contact
yes
enter name and number
vikas 7893572744
do you want to create another contact
no
```



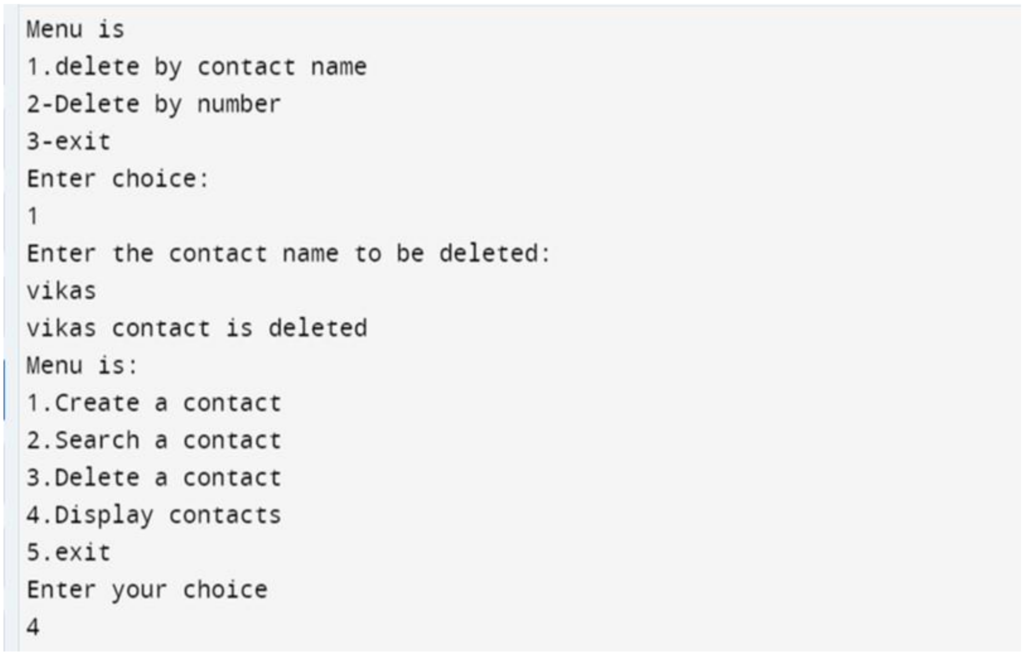
```
Menu is:
1.Create a contact
2.Search a contact
3.Delete a contact
4.Display contacts
5.exit
Enter your choice
4
menu:
1-display according to numbers
2-display according to names
3-exit
Enter your choice
2
a
aishwarya->9014964314
b
bhargav->9347786796
```



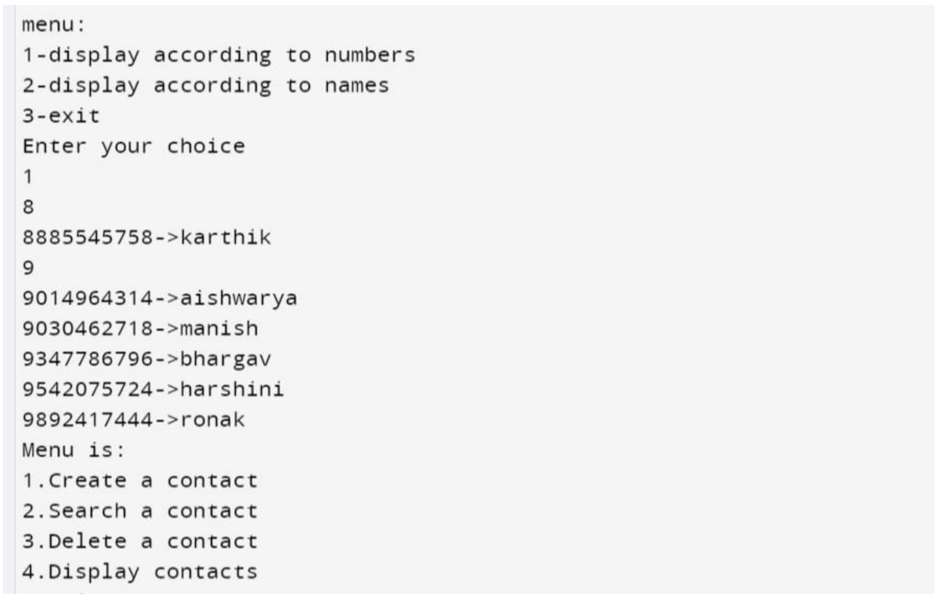
```
h
harshini->9542075724
k
karthik->8885545758
m
manish->9030462718
r
ronak->9892417444
v
vikas->7893572744
Menu is:
1.Create a contact
2.Search a contact
3.Delete a contact
4.Display contacts
5.exit
Enter your choice
2
```



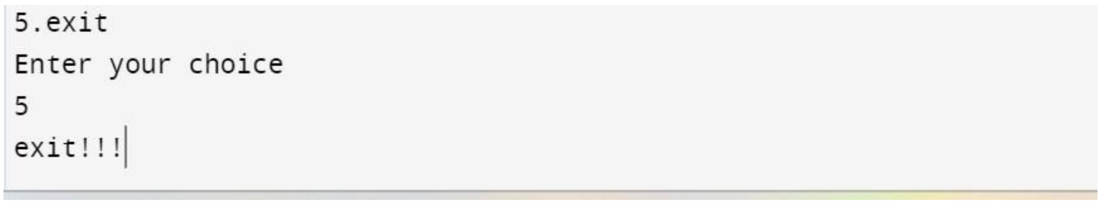
```
Menu is :
1.search by first letter of the name
2.search by full name
3-Search by first digit of number
4-search by full number
press '-1' to exit
1
Enter a letter:
m
Available Contacts starting with given letter is
manish  9030462718
Menu is:
1.Create a contact
2.Search a contact
3.Delete a contact
4.Display contacts
5.exit
Enter your choice
3
```



```
Menu is
1.delete by contact name
2-Delete by number
3-exit
Enter choice:
1
Enter the contact name to be deleted:
vikas
vikas contact is deleted
Menu is:
1.Create a contact
2.Search a contact
3.Delete a contact
4.Display contacts
5.exit
Enter your choice
4
```



```
menu:
1-display according to numbers
2-display according to names
3-exit
Enter your choice
1
8
8885545758->karthik
9
9014964314->aishwarya
9030462718->manish
9347786796->bhargav
9542075724->harshini
9892417444->ronak
Menu is:
1.Create a contact
2.Search a contact
3.Delete a contact
4.Display contacts
```



```
5.exit
Enter your choice
5
exit!!!
```

CHAPTER 5 - CONCLUSION

The "Phone Directory Management System" project holds considerable potential for future enhancements and expansions, positioning itself as a dynamic and versatile tool for efficient contact management. With the ever-evolving landscape of technology and user needs, there are several avenues for extending the scope of this project. One potential avenue for future development is the integration of advanced search functionalities, such as fuzzy searching or natural language processing. This would enable users to find contacts even when the input does not exactly match the stored data, enhancing the system's usability. Additionally, integration with online databases or APIs could provide real-time updates for contact details, ensuring accuracy and currency of information.

Another promising direction involves the implementation of a graphical user interface (GUI), transforming the project into a more visually appealing and interactive application. A GUI could offer features like drag-and-drop contact management, customizable contact cards, and visual representations of contact relationships. As mobile devices continue to dominate communication, the adaptation of the project into a mobile app could further expand its reach. Users could access and manage their contacts on the go, leveraging features like QR code sharing or location-based reminders for contacts.

FUTURE SCOPE

By integrating advanced contact categorization features, users can organize contacts more efficiently using user-defined tags or groups. Additionally, enabling integration with external services like cloud storage or mobile synchronization would enhance data accessibility and flexibility. Improving the user interface with graphical elements or transitioning to a GUI-based application would enhance usability. Features such as automatic data backup, advanced search options, multi-user support with role-based access, and localization for different languages would broaden the system's appeal and utility. Incorporating analytics tools for data insights and customizable reporting would further elevate the system's functionality. These future enhancements promise to transform the Phone Directory Management System into a sophisticated and comprehensive solution, meeting diverse contact management needs and ensuring continued relevance and usefulness in various user scenarios.

CHAPTER 6 - REFERENCES

[1]. Data Structures through C.

[2].https://www.bing.com/ck/a?!&&p=cdf81726bb30926bJmltdHM9MTY5MTQ1MjgwMCZpZ3VpZD0xZGM2NzlhYi0wNzliLTY5N2ItMTRmYS02YWVlMDY3MzY4NmYmaW5zaWQ9NTMyOQ&ptn=3&hsh=3&fclid=1dc679ab-079b-697b-14fa-6aee0673686f&psq=phone+directory+system+in+data+structure+in+c&u=a1aHR0cHM6Ly93d3cuc2xpZGVzaGFyZS5uZXQvdGhhbmtzZm9ydmlzaXRpbmdoZXJlL3RlbGVwaG9uZS1kaXJlY3RvcnktaW4tYw&ntb=1

[3].https://www.bing.com/ck/a?!&&p=ac48f8013617a59aJmltdHM9MTY5MTQ1MjgwMCZpZ3VpZD0xZGM2NzlhYi0wNzliLTY5N2ItMTRmYS02YWVlMDY3MzY4NmYmaW5zaWQ9NTI3Mg&ptn=3&hsh=3&fclid=1dc679ab-079b-697b-14fa-6aee0673686f&psq=phone+directory+system+in+data+structure+in+c+algorithms&u=a1aHR0cHM6Ly9jb2RlcmV2aWV3LnN0YWNrZXhjaGFuZ2UuY29tL3F1ZXN0aW9ucy8xMTU4ODAvcGhvbmVib29rLWltcGxlbWVudGF0aW9uLWluLWM&ntb=1