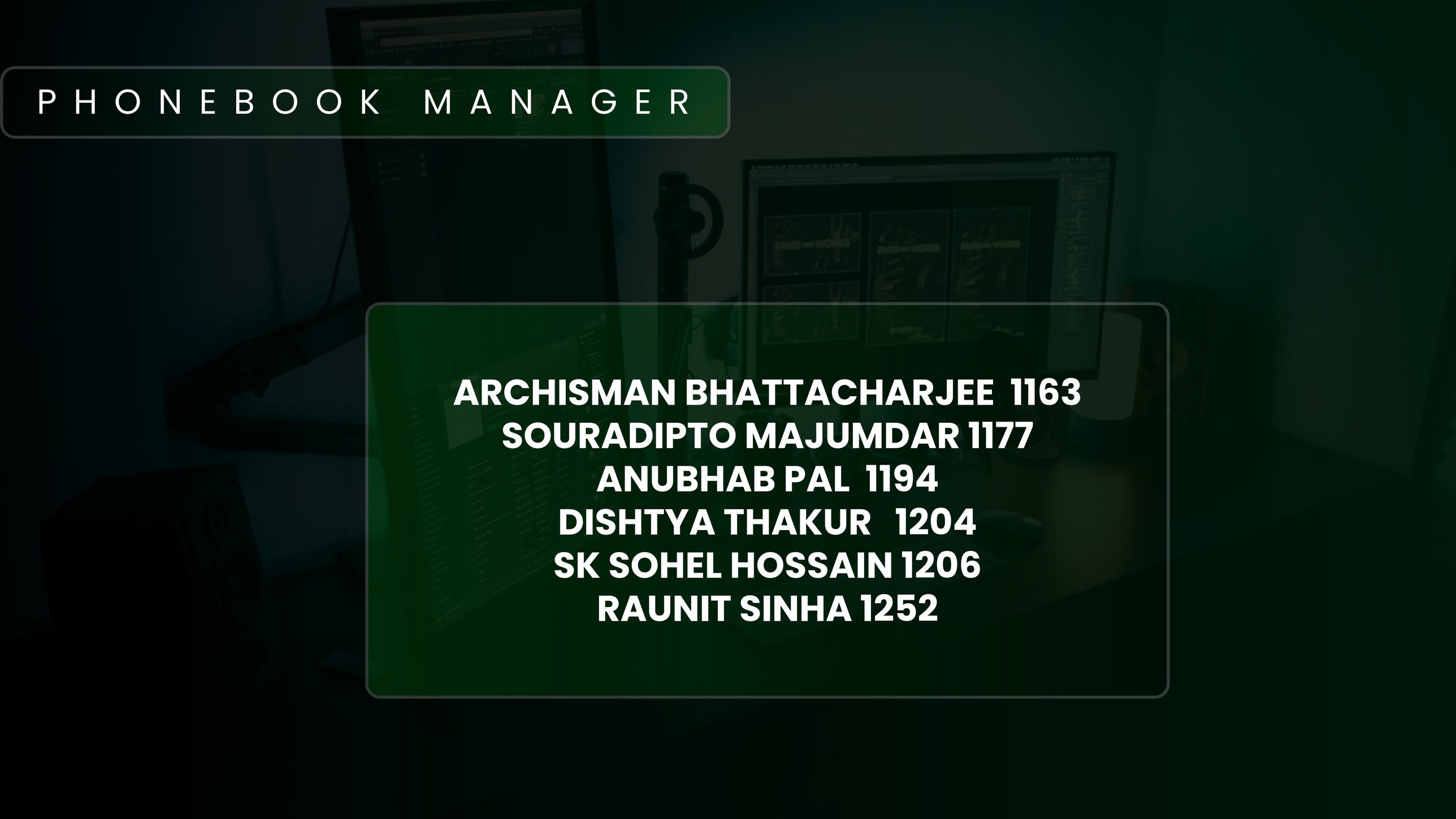
# PHONEBOOK MANAGER



**ARCHISMAN BHATTACHARJEE 1163**
**SOURADIPTO MAJUMDAR 1177**
**ANUBHAB PAL 1194**
**DISHTYA THAKUR 1204**
**SK SOHEL HOSSAIN 1206**
**RAUNIT SINHA 1252**

# Introduction

>This program is a simple Contact Management System designed to store, manage, and retrieve contact information.
>It's implemented in C and utilizes a Binary Search Tree (BST) for efficient data organization and retrieval.

This provides advantages like faster searching compared to linear structures.


# About Us

We have created a code to build a phone book where you can use a BST data structure to store people's information, such as their names and phone numbers. This application will help you add, view, edit, and delete contacts, and also provide functionality for searching contacts based on various criteria, like names

This is a Phonebook Manager written in C. It allows users to:

- Add contacts
- Edit contacts
- Delete contacts
- Search contacts (by name or phone)
- Display all contacts
- Save/load from a file (contacts.txt)

It uses a Binary Search Tree (BST) to store and manage the contacts efficiently, sorted alphabetically by name.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX_NAME 100
#define MAX_PHONE 20

typedef struct Contact
{
    char name[MAX_NAME];
    char phone[MAX_PHONE];
    struct Contact* left;
    struct Contact* right;
} Contact;
```

The structure contact consists of a name and phone number.
It also consists of  left and right pointers to form the BST based on name (lexicographically).

Each node in the BST represents one contact, storing:
- `name`: Name of the person (string)
- `phone`: Phone number (string)
- `left`: Pointer to left subtree (contacts with smaller names)
- `right`: Pointer to right subtree (contacts with larger names)

Core Functions

**1. `capitalizeEachWord(char* str)`**

Purpose: Formats the name by capitalizing the
 **first letter of each word** and making the rest lowercase.
Input: "jOHn DOe"
Output: "John Doe"

```
void capitalizeEachWord(char* str)
{
    int inWord = 0;
    for (int i = 0; str[i]; i++)
    {
        if (isspace(str[i]))
        {
            inWord = 0;
        }
        else
        {
            if (!inWord)
            {
                str[i] = toupper(str[i]);
                inWord = 1;
            }
            else
            {
                str[i] = tolower(str[i]);
```

## 2. createNode(char* name, char* phone)

Creates a new contact node in memory and initializes fields.

Returns a pointer to the new contact node.

```
Contact* createNode(char* name, char* phone)
{
    Contact* newNode = (Contact*)malloc(sizeof(Contact));
    strcpy(newNode->name, name);
    strcpy(newNode->phone, phone);
    newNode->left = newNode->right = NULL;
    return newNode;
}
```

## 3. insert(Contact* root, char* name, char* phone)

- Inserts a new contact in BST order (by name).
- If the tree is empty, it returns a new node.
- If the name is less than root's name → insert to left.
- If greater → insert to right.
- If equal → prints message (no duplicates allowed).
- This maintains the alphabetical ordering of contacts.

```
Contact* insert(Contact* root, char* name, char* phone)
{
    if (root == NULL)
        return createNode(name, phone);
    int cmp = strcmp(name, root->name);
    if (cmp < 0)
        root->left = insert(root->left, name, phone);
    else if (cmp > 0)
        root->right = insert(root->right, name, phone);
    else
        printf("Contact '%s' already exists.\n", name);
    return root;
}
```

**4. search(Contact* root, char* name)**

Searches the BST by name. Returns pointer to the contact if found, or NULL.
It uses standard BST logic:
If name matches → return
If name < root → go left
If name > root → go right

```
Contact* search(Contact* root, char* name)
{
    if (!root || strcmp(name, root->name) == 0)
        return root;
    if (strcmp(name, root->name) < 0)
        return search(root->left, name);
    else
        return search(root->right, name);
}
```

**5. searchByPhone(Contact* root, const char* phone)**

Searches the entire tree for a phone number.

Since the tree is sorted by name (not phone), it uses recursive traversal:

Check current node

Recurse into left and right

Slower than name search (O(n)), but necessary.

```
Contact* searchByPhone(Contact* root, const char* phone)
{
    if (!root)
        return NULL;
    if (strcmp(root->phone, phone) == 0)
        return root;
    Contact* left = searchByPhone(root->left, phone);
    if (left) return left;
    return searchByPhone(root->right, phone);
}
```

## 6. suggestMatches(Contact* root, const char* prefix)

Suggests contacts whose name or phone starts with the given prefix. Useful when exact search fails.

This performs a prefix match and suggests possible matching contacts whose name or number start with the
given prefix. This is useful for:

- Typo correction
- Partial input
- Searching with incomplete info

```
void suggestMatches(Contact* root, const char* prefix)
{
    if (!root)
        return;
    if (strncmp(root->name, prefix, strlen(prefix)) == 0 || strncmp(root->phone, prefix, strlen(prefix)) == 0)
    {
        printf("%s: %s\n", root->name, root->phone);
    }
    suggestMatches(root->left, prefix);
    suggestMatches(root->right, prefix);
}
```

The function **Contact* minValueNode(Contact* node)** is a helper function used in the deletion process of a Binary Search Tree (BST).
It finds the node with the smallest (minimum) name value in a given subtree — typically, the leftmost node.
It starts with the node provided as input, traverses left as far as possible using current=current-> left and when it reaches a node with no left child, the node is returned.

```
Contact* minValueNode(Contact* node)
{
    Contact* current = node;
    while (current && current->left)
        current = current->left;
    return current;
}
```

```
Contact* delete(Contact* root, char* name)
{
    if (!root) return root;
    int cmp = strcmp(name, root->name);
    if (cmp < 0)
        root->left = delete(root->left, name);
    else if (cmp > 0)
        root->right = delete(root->right, name);
    else {
        if (!root->left)
        {
            Contact* temp = root->right;
            free(root);
            return temp;
        }
        else if (!root->right)
        {
            Contact* temp = root->left;
            free(root);
            return temp;
        }
        Contact* temp = minValueNode(root->right);
        strcpy(root->name, temp->name);
        strcpy(root->phone, temp->phone);
        root->right = delete(root->right, temp->name);
    }
    return root;
}
```

**7. delete(Contact* root, char* name)**

Removes a contact by name from the tree.
It follows 3 main cases:
Case 1: No children → free node
Case 2: One child → replace with that child
Case 3: Two children:
Find in-order successor (smallest value in right subtree)
Replace current node with successor
Delete the successor node
This ensures the BST structure is preserved.

**8. edit(Contact** root, char* name)**

Lets the user edit a contact:
If only name changes → delete and reinsert (because BST key changes)
If only phone → directly update phone field
If both → same as name change (delete & insert)
This handles BST integrity and updates cleanly.

```
void edit(Contact** root, char* name)
{
    Contact* node = search(*root, name);
    if (node)
    {
        int choice;
        char newName[MAX_NAME], newPhone[MAX_PHONE];
        printf("Edit Options:\n1. Name\n2. Phone\n3. Both\nEnter choice: ");
        scanf("%d", &choice);
        getchar();

        switch (choice)
        {
            case 1:
                printf("Enter new name: ");
                fgets(newName, MAX_NAME, stdin);
                newName[strcspn(newName, "\n")] = 0;
                capitalizeEachWord(newName);
                strcpy(newPhone, node->phone);
                *root = delete(*root, node->name);
                *root = insert(*root, newName, newPhone);
                printf("Name updated.\n");
                break;
```

```
case 2:
    printf("Enter new phone: ");
    fgets(newPhone, MAX_PHONE, stdin);
    newPhone[strcspn(newPhone, "\n")] = 0;
    strcpy(node->phone, newPhone);
    printf("Phone updated.\n");
    break;
```

```
            case 3:
                printf("Enter new name: ");
                fgets(newName, MAX_NAME, stdin);
                newName[strcspn(newName, "\n")] = 0;
                capitalizeEachWord(newName);

                printf("Enter new phone: ");
                fgets(newPhone, MAX_PHONE, stdin);
                newPhone[strcspn(newPhone, "\n")] = 0;

                *root = delete(*root, node->name);
                *root = insert(*root, newName, newPhone);
                printf("Name and phone updated.\n");
                break;

            default:
                printf("Invalid option.\n");
        }
    }
    else
    {
        printf("Contact '%s' not found.\n", name);
    }
```

**9. display(Contact* root)**

In-order traversal to print all contacts alphabetically.
This function is responsible for printing all the contacts in the phonebook in alphabetical order by name.

**display(root->left);**

- Visit the left subtree (names that are alphabetically smaller).

**printf("%s: %s\n", root->name, root->phone);**

- Process the current node (print the contact).

**display(root->right);**

- Visit the right subtree (names that are alphabetically greater).

This ensures names are printed in sorted order.

```
void display(Contact* root)
{
    if (root)
    {
        display(root->left);
        printf("%s: %s\n", root->name, root->phone);
        display(root->right);
    }
}
```

# File Handling Functions:

**loadFromFile(const char* filename, Contact* root)**

This reads from contacts.txt line-by-line.
It:

- Parses each line
- Capitalizes name
- Inserts into BST

Returns the updated tree.

```
Contact* loadFromFile(const char* filename, Contact* root)
{
    FILE* file = fopen(filename, "r");
    if (!file) return root;
    char line[200], name[MAX_NAME], phone[MAX_PHONE];
    while (fgets(line, sizeof(line), file))
    {
        if (sscanf(line, "%[^,],%s", name, phone) == 2)
        {
            capitalizeEachWord(name);
            root = insert(root, name, phone);
        }
    }
    fclose(file);
    return root;
}
```

## saveToFile(FILE* file, Contact* root)

Saves contacts to file using in-order traversal.

Ensures alphabetical order in file

```
void saveToFile(FILE* file, Contact* root)
{
    if (root)
    {
        saveToFile(file, root->left);
        fprintf(file, "%s,%s\n", root->name, root->phone);
        saveToFile(file, root->right);
    }
}
```

# User Interface – main() function:

Menu:
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit

The program loads data at start and enters a loop where the user can perform operations.

```c
int main()
{
    Contact* root = NULL;
    char name[MAX_NAME], phone[MAX_PHONE];
    int choice;

    root = loadFromFile("contacts.txt", root);

    while (1)
    {
        printf("-----Phonebook Manager-----");
        printf("\n1. Add Contact\n2. Edit Contact\n3. Delete Contact\n4. Search Contact\n5. Display All\n6. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        getchar();

        switch (choice)
        {
            case 1:
                printf("Enter name: ");
                fgets(name, MAX_NAME, stdin);
                name[strcspn(name, "\n")] = 0;
                capitalizeEachWord(name);
                printf("Enter phone: ");
                fgets(phone, MAX_PHONE, stdin);
                phone[strcspn(phone, "\n")] = 0;
                root = insert(root, name, phone);
                printf("\n");
                break;

            case 2:
                printf("Enter name to edit: ");
                fgets(name, MAX_NAME, stdin);
                name[strcspn(name, "\n")] = 0;
                capitalizeEachWord(name);
                edit(&root, name);
                printf("\n");
                break;
```

```c
case 3:
    printf("Enter name to delete: ");
    fgets(name, MAX_NAME, stdin);
    name[strcspn(name, "\n")] = 0;
    capitalizeEachWord(name);
    root = delete(root, name);
    printf("Contact deleted\n");
    printf("\n");
    break;

case 4:
    printf("Enter name or phone to search: ");
    fgets(name, MAX_NAME, stdin);
    name[strcspn(name, "\n")] = 0;
    capitalizeEachWord(name);

    Contact* found = search(root, name);
    if (found)
    {
        printf("Found: %s - %s\n", found->name, found->phone);
        printf("1. Edit\n2. Delete\n3. Do Nothing\nEnter choice: ");
        int subchoice;
        scanf("%d", &subchoice);
        getchar();
        switch (subchoice)
        {
            case 1:
            edit(&root, found->name);
            break;

            case 2:
            root = delete(root, found->name);
            printf("Contact deleted\n");
            break;

            case 3:
            break;

            default:
            printf("Invalid choice\n");
        }
    }
    else
    {
        Contact* phoneMatch = searchByPhone(root, name);
        if (phoneMatch)
        {
            printf("Found by phone: %s - %s\n", phoneMatch->name, phoneMatch->phone);
        }
        else
        {
            printf("Contact not found. Suggestions starting with '%s':\n", name);
            suggestMatches(root, name);
            printf("Do you want to add this contact? (Y/N): ");
            char ans;
            scanf("%c", &ans);
            getchar();
            ans = toupper(ans);
            if (ans == 'Y')
            {
                strcpy(name, name);
                capitalizeEachWord(name);
                printf("Enter phone: ");
                fgets(phone, MAX_PHONE, stdin);
                phone[strcspn(phone, "\n")] = 0;
                root = insert(root, name, phone);
                printf("\n");
            }
            else if(ans == 'N')
            {
                printf("Contact not added.\n");
            }
            else
            {
                printf("Invalid Input! Contact not added.\n");
            }
        }
    }
    break;

case 5:
    printf("All Contacts:\n");
    display(root);
    break;
```

```
            case 6:
                printf("Saving and exiting...\n");
                FILE* file = fopen("contacts.txt", "w");
                if (file)
                {
                    saveToFile(file, root);
                    fclose(file);
                    printf("Contacts saved to contacts.txt\n");
                }
                else
                {
                    printf("Error saving file.\n");
                }
                return 0;

            default:
                printf("Invalid choice.\n");
        }
    }
    return 0;
}
```

# OUTPUTS

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 1
Enter name: Archishman Bhattacharjee
Enter phone: 877397664
```

**ADD CONTACTS**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 5
All Contacts:
Anubhab Pal: 1231449654
Archishman Bhattacharjee: 8777397554
Dishtya Thakur: 1154613214
Raunit Sinha: 196523168
Shiven Gupta: 479869729
Shiven Thakker: 2314546651
Shubhadip Mandal: 1676441879
Sk Sohek Hossain: 1321164123
Souradipto Majumdar: 3164661185
-----Phonebook Manager-----
```

**DISPLAY ALL CONTACTS IN THE PHONEBOOK**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 2
Enter name to edit: Souradipto Mukherjee
Edit Options:
1. Name
2. Phone
3. Both
Enter choice: 1
Enter new name: Souradipto Majumdar
Name updated.
```

**EDIT OPTION –NAME UPDATION**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 2
Enter name to edit: Archishman Bhattacharjee
Edit Options:
1. Name
2. Phone
3. Both
Enter choice: 2
Enter new phone: 8777397554
Phone updated.
```

**EDIT OPTION –PHONE NUMBER UPDATION**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 2
Enter name to edit: adt brmn
Edit Options:
1. Name
2. Phone
3. Both
Enter choice: 3
Enter new name: Aditi Barman
Enter new phone: 1465131343
Name and phone updated.
```

**EDIT OPTION – NAME AND PHONE NUMBER BOTH UPDATION**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 4
Enter name or phone to search: Archishman Bhattacharjee
Found: Archishman Bhattacharjee - 8777397554
1. Edit
2. Delete
3. Do Nothing
Enter choice: 1
Edit Options:
1. Name
2. Phone
3. Both
Enter choice: 2
Enter new phone: 8777397554
Phone updated.
```

**SEARCH BY NAME  AND UPDATION OF NUMBER**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 4
Enter name or phone to search: Archishman Bhattacharjee
Found: Archishman Bhattacharjee - 3154313456
1. Edit
2. Delete
3. Do Nothing
Enter choice: 1
Edit Options:
1. Name
2. Phone
3. Both
Enter choice: 3
Enter new name: Archishman
Enter new phone: 8777397664
Name and phone updated.
```

**SEARCHING AND THEN EDITING BOTH NAME AND NUMBER**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 3
Enter name to delete: Shiven Gupta
```

**DELETION OF A CONTACT**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 4
Enter name or phone to search: Aditi Barman
Found: Aditi Barman - 1465131343
1. Edit
2. Delete
3. Do Nothing
Enter choice: 2
Contact deleted
```

**SEARCHING PHONE NUMBER AND THEN DELETION**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 5
All Contacts:
Anubhab Pal: 1231449654
Archishman Bhattacharjee: 8777397554
Dishtya Thakur: 1154613214
Raunit Sinha: 196523168
Shiven Thakker: 2314546651
Shubhadip Mandal: 1676441879
Sk Sohek Hossain: 1321164123
Souradipto Majumdar: 3164661185
```

**DISPLAYING ALL CONTACTS AFTER DELETION**

```
-----Phonebook Manager-----
1. Add Contact
2. Edit Contact
3. Delete Contact
4. Search Contact
5. Display All
6. Exit
Enter choice: 5
All Contacts:
Anubhab Pal: 1231449654
Archishman: 8777397664
Dishtya Thakur: 1154613214
Raunit Sinha: 196523168
Shiven Thakker: 2314546651
Shubhadip Mandal: 1676441879
Sk Sohek Hossain: 1321164123
Souradipto Majumdar: 3164661185
```

**FINAL DISPLAY AFTER SEARCHING, EDITING, UPADTING AND DELETING NUMBERS FROM A PHONEBOOK**