SCHOOL OF COMPUTING

BACHELOR OF COMPUTER SCIENCE ENGINEERING

LOVELY PROFESSIONAL UNIVERSITY, PHAGWARA, PUNJAB, INDIA.

FINAL REPORT ON PHONEBOOK MANAGEMENT SYSTEM

CSE101: COMPUTER PROGRAMMING

DELIVERED BY:		RECEIVED BY:	
NAME OF THE STUDENT:	Arpan Gupta	NAME OF THE FACULTY:	Dr. Jyoti Godara
REG. NO.:	12210528	UID:	18190
ROLL NO & SECTION:	12 & K22BN	SIGNATURE	

DESCRIPTION: -

This code is a simple phonebook management system written in C. It allows the user to add new contacts to a phonebook, view all contacts in the phonebook, and search for a specific contact by name. The phonebook is stored in a file named phonebook.txt.
The code uses several modules from the C standard library. The stdio.h module is used for input/output operations such as reading from and writing to files and standard input/output streams. The string.h module is used for string manipulation functions such as sscanf. The program also defines a contact struct to represent each contact in the phonebook and uses functions such as add_contact, print_phonebook, and search_contact to implement the phonebook management functionality.

MODULE EXPLANATION: -

1. **add_contact**: The add_contact function is used to add a new contact to the phonebook. It first defines a new contact struct named new_contact and prompts the user to enter the name and phone number of the new contact. The entered values are stored in the name and phone_number fields of the new_contact struct. The function then opens the phonebook.txt file in append mode and writes the new contact to the file using the fprintf function. Finally, it closes the file using the fclose function and prints a message to indicate that the contact was added successfully.

2. **print_phonebook:** The print_phonebook function is used to print all the contacts in the phonebook. It first opens the phonebook.txt file in read mode and checks if the file exists. If it does not exist, the function prints a message to indicate that the phonebook is empty and returns. Otherwise, it prints a header and reads each line from the file using the fgets function. For each line, it extracts the name and phone number of the contact using the sscanf function and prints them to the screen using printf. The function also increments the num_contacts variable to keep track of how many contacts have been printed. Finally, it closes the file using the fclose function.
3. **search_contact:** The search_contact function is used to search for a specific contact in the phonebook by name. It first opens the phonebook.txt file in read mode and checks if the file exists. If it does not exist, the function prints a message to indicate that the phonebook is empty and returns. The function then prompts the user to enter a name to search for and stores it in the search_name variable. It then reads each line from the file using the fgets function and extracts the name and phone number of the contact using the sscanf function. The function compares the extracted name with the search name using the strcmp function. If they are equal, it means that a matching contact has been found and it prints the contact to the screen using printf. Finally, it closes the file using the fclose function.

4. **delete_contact:** The delete_contact function is used to delete a specific contact from the phonebook by name. It first opens the phonebook.txt file in read mode and checks if the file exists. If it does not exist, the function prints a message to indicate that the phonebook is

empty and returns. The function then prompts the user to enter a name to delete and stores it in the delete_name variable. It then opens a temporary file named temp.txt in write mode and checks if it can be created. If it cannot be created, the function prints an error message and returns. The function then reads each line from the phonebook.txt file using the fgets function and extracts the name and phone number of the contact using the sscanf function. It compares the extracted name with the delete name using the strcmp function. If they are equal, it means that this is the contact to be deleted and sets a flag named contact_deleted to 1. It does not write this contact to the temporary file. Otherwise, if the names are not equal, it writes this contact to the temporary file using the fprintf function. When all lines have been read and processed, the function closes both files using the fclose function. If no contact was deleted, it means that no matching contact was found and prints a message to indicate that no contact was found and deletes the temporary file using the remove function. Otherwise, if a contact was deleted, it means that a matching contact was found and deleted from the temporary file.

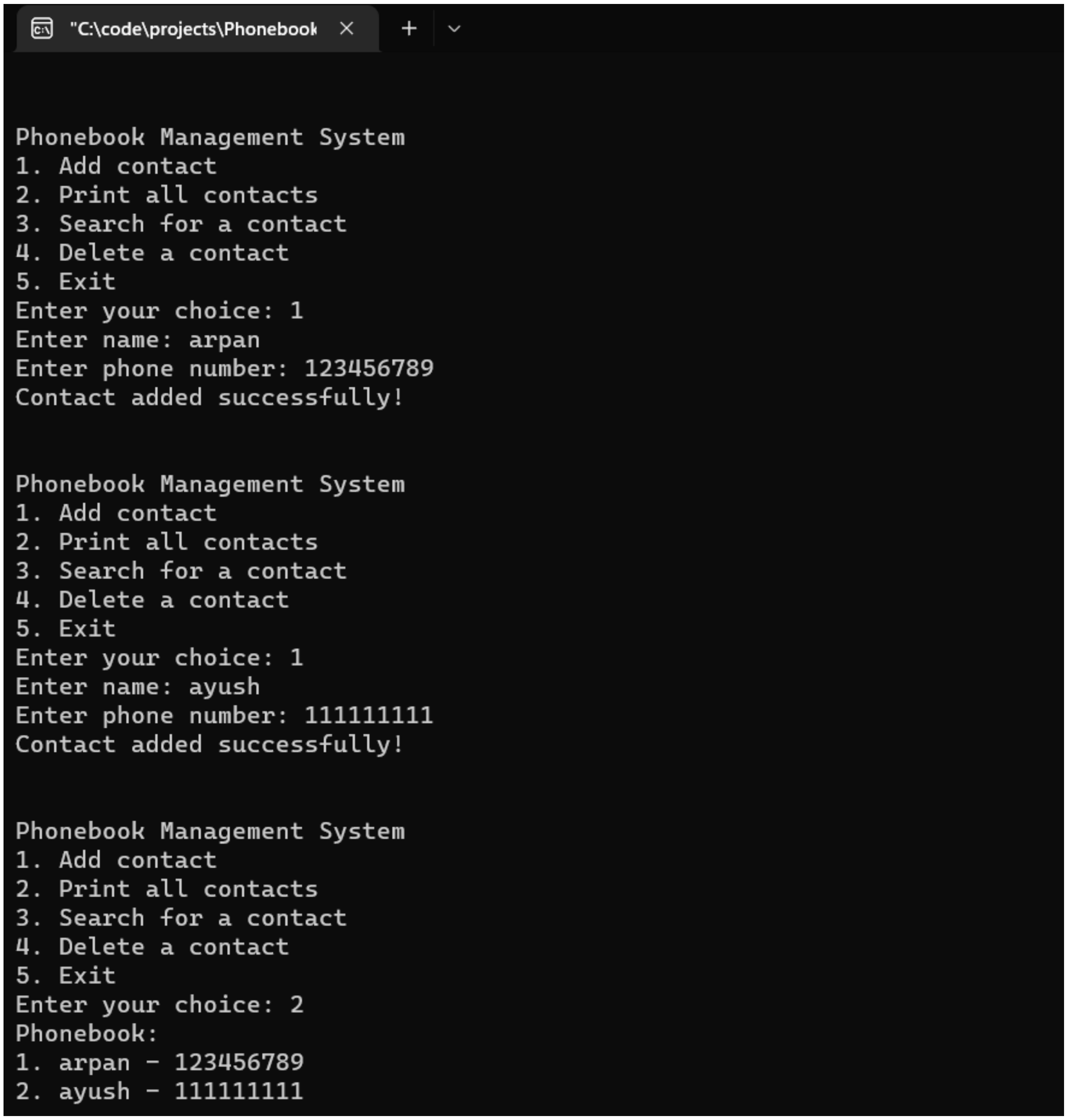
5. **Main:** The main function of the phonebook management system provides a menu-driven interface for the user to interact with the phonebook. It defines an int variable named choice to store the user's menu choice and enters a do-while loop that continues until the user chooses to exit the program. At the beginning of each iteration of the loop, the function prints a menu of options and prompts the user to enter their choice. The function then uses a switch statement to execute different code depending on the value of choice. The options include adding a contact, printing all contacts, searching for a contact, deleting a contact, and exiting the program. When the loop exits, the function returns 0 to indicate successful completion.

SOURCE CODE : -

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

// Define a struct to represent each contact in the phonebook
struct contact {
    char name[50];
    char phone_number[20];
};

// Keep track of the number of contacts in the phonebook
int num_contacts = 0;

// Function to add a contact to the phonebook
void add_contact() {
    struct contact new_contact;
    printf("Enter name: ");
    scanf("%s", new_contact.name);
    printf("Enter phone number: ");
    scanf("%s", new_contact.phone_number);

    // Open the phonebook file in append mode
    FILE *fp = fopen("phonebook.txt", "a");

    // Write the new contact to the file
    fprintf(fp, "%s %s\n", new_contact.name,
new_contact.phone_number);

    // Close the file
    fclose(fp);

    printf("Contact added successfully!\n");
}

// Function to print all contacts in the phonebook
void print_phonebook() {
    // Open the phonebook file in read mode
    FILE *fp = fopen("phonebook.txt", "r");

    if (fp == NULL) {
        printf("Phonebook is empty.\n");
        return;
    }

    printf("Phonebook:\n");

    // Read each line from the file and print the contact
    char line[100];
    while (fgets(line, 100, fp) != NULL) {
        char name[50];
        char phone_number[20];
        sscanf(line, "%s %s", name, phone_number);
        printf("%d. %s - %s\n", ++num_contacts, name,
phone_number);
    }

    // Close the file
    fclose(fp);
}

// Function to search for a contact in the phonebook
void search_contact() {
    // Open the phonebook file in read mode
    FILE *fp = fopen("phonebook.txt", "r");

    if (fp == NULL) {
        printf("Phonebook is empty.\n");
        return;
    }

    char search_name[50];
    printf("Enter name to search for: ");
    scanf("%s", search_name);

    // Read each line from the file and search for the contact
    char line[100];
    while (fgets(line, 100, fp) != NULL) {
        char name[50];
        char phone_number[20];
        sscanf(line, "%s %s", name, phone_number);
        if (strcmp(name, search_name) == 0) {
            printf("%s - %s\n", name, phone_number);
            return;
        }
    }

    printf("Contact not found.\n");

    // Close the file
    fclose(fp);
}
// Function to delete a contact from the phonebook
// Function to delete a contact from the phonebook
    void delete_contact() {
        // Open the phonebook file in read mode
        FILE *fp = fopen("phonebook.txt", "r");

        if (fp == NULL) {
            printf("Phonebook is empty.\n");
            return;
        }

        char delete_name[50];
        printf("Enter name to delete: ");
        scanf("%s", delete_name);

        // Open a temporary file in write mode
        FILE *temp_fp = fopen("temp.txt", "w");

        if (temp_fp == NULL) {
            printf("Error creating temporary file.\n");
            return;
        }

        // Read each line from the file and copy to the
temporary file except the one to be deleted
        char line[100];
        int contact_deleted = 0;
        while (fgets(line, 100, fp) != NULL) {
            char name[50];
            char phone_number[20];
            sscanf(line, "%s %s", name, phone_number);
            if (strcmp(name, delete_name) == 0) {
                contact_deleted = 1;
            } else {
                fprintf(temp_fp, "%s %s\n", name, phone_number);
            }
        }

        // Close the files
        fclose(fp);
        fclose(temp_fp);

        if (!contact_deleted) {
            printf("Contact not found.\n");
            // Delete the temporary file
            remove("temp.txt");
        } else {
            // Delete the original file
            remove("phonebook.txt");
            // Replace it with the temporary file
            rename("temp.txt", "phonebook.txt");
            printf("Contact deleted successfully!\n");
        }
    }



int main() {
    int choice;
    do {
        printf("\n\n");
        printf("Phonebook Management System\n");
        printf("1. Add contact\n");
        printf("2. Print all contacts\n");
        printf("3. Search for a contact\n");
        printf("4. Delete a contact\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
```

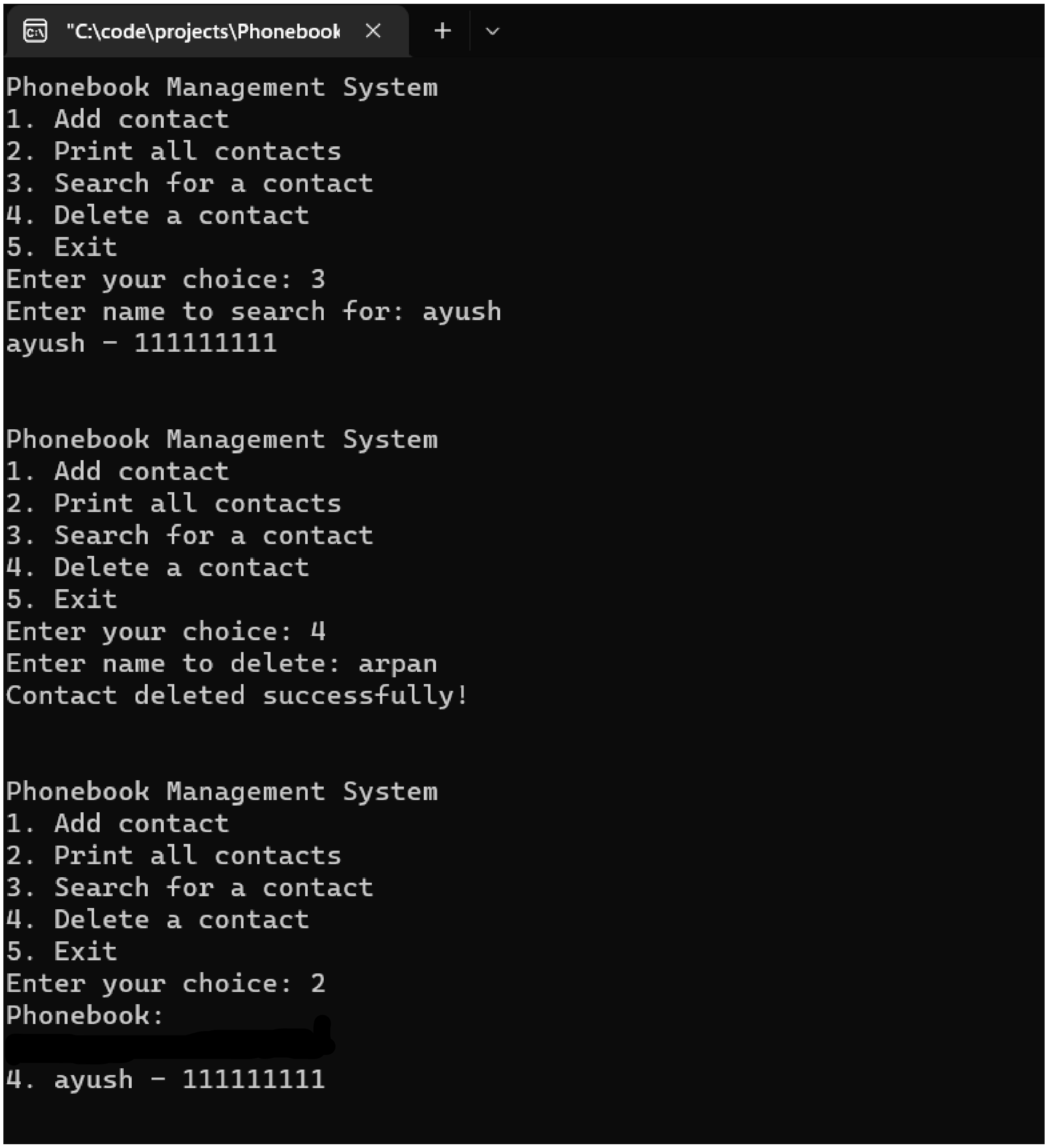
```c
            case 1:
                add_contact();
                break;
            case 2:
                print_phonebook();
                break;
            case 3:
                search_contact();
                break;
            case 4:
                delete_contact();
                break;
            case 5:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice.\n");
                break;
        }
    } while (choice != 5);

    return 0;
}
```

OUPUT : -

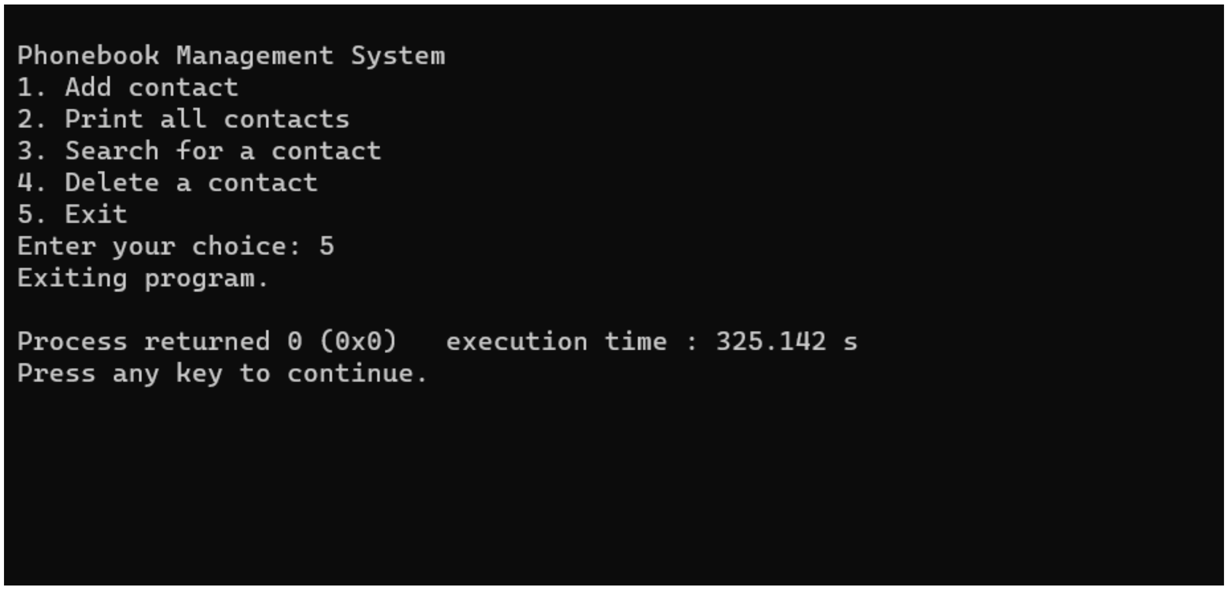


"C:\code\projects\Phonebook

```
Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 1
Enter name: arpan
Enter phone number: 123456789
Contact added successfully!


Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 1
Enter name: ayush
Enter phone number: 111111111
Contact added successfully!


Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 2
Phonebook:
1. arpan - 123456789
2. ayush - 111111111
```


"C:\code\projects\Phonebook

```
Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 3
Enter name to search for: ayush
ayush - 111111111


Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 4
Enter name to delete: arpan
Contact deleted successfully!


Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 2
Phonebook:
[redacted]
4. ayush - 111111111
```



```
Phonebook Management System
1. Add contact
2. Print all contacts
3. Search for a contact
4. Delete a contact
5. Exit
Enter your choice: 5
Exiting program.

Process returned 0 (0x0)   execution time : 325.142 s
Press any key to continue.
```

Thank You