UNIVERSITY INSTITUDE OF

COMPUTING

SUBJECT: COMPUTER PROGRAMMING

PROJECT TITLE: SIMPLE PROJECT ADDRESS

GROUP PROJECT

GROUP NAME: REBEL

1.NAME : PRABHAT KUMAR

UID: 24BCD10072

2. NAME: MANJINDER SINGH

UID: 24BCD10081

SUBMITTED TO:

SHUCHI SHARMA

Simple Address Book (C Program)

Project Overview:

This application allows users to manage contacts, including adding, deleting, updating, and searching for contacts. Contacts are stored with their name, phone number, and email address.

C Code Implementation:

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

#define MAX_CONTACTS 100
// Structure to hold contact information
typedef struct {
    char name[50];
    char phone[20];
    char email[50];
} Contact;

// Function declarations
void addContact(Contact contacts[], int *count);
```

```
void deleteContact(Contact contacts[], int *count);
void updateContact(Contact contacts[], int count);
void searchContact(Contact contacts[], int count);
void displayContacts(Contact contacts[], int count);
int getMenuChoice();

int main() {
    Contact contacts[MAX_CONTACTS];
    int contactCount = 0;
    int choice;
  while (1) {
        choice = getMenuChoice();
 switch (choice) {
            case 1:
                addContact(contacts, &contactCount);
                break;
            case 2:
                deleteContact(contacts, &contactCount);
                break;
            case 3:
```

```
            updateContact(contacts, contactCount);
            break;
        case 4:
            searchContact(contacts, contactCount);
            break;
        case 5:
            displayContacts(contacts, contactCount);
            break;
        case 6:
            printf("Exiting program...\n");
            return 0;
        default:
            printf("Invalid choice! Please try again.\n"); }
    }
 return 0;
}
// Function to display menu and get user choice
int getMenuChoice() {
    int choice;
    printf("\nSimple Address Book\n");
```

```
    printf("1. Add Contact\n");
    printf("2. Delete Contact\n");
    printf("3. Update Contact\n");
    printf("4. Search Contact\n");
    printf("5. Display Contacts\n");
    printf("6. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    return choice;
}
// Function to add a new contact
void addContact(Contact contacts[], int *count) {
    if (*count >= MAX_CONTACTS) {
        printf("Error: Maximum number of contacts reached.\n");
        return;
    }
 printf("Enter contact name: ");
    getchar(); // Clear newline character left by previous input
    fgets(contacts[*count].name,
sizeof(contacts[*count].name), stdin);
```

```
    contacts[*count].name[strcspn(contacts[*count].name,
"\n")] = 0; // Remove newline
 printf("Enter contact phone number: ");
    fgets(contacts[*count].phone,
sizeof(contacts[*count].phone), stdin);
    contacts[*count].phone[strcspn(contacts[*count].phone,
"\n")] = 0; // Remove newline
 printf("Enter contact email: ");
    fgets(contacts[*count].email,
sizeof(contacts[*count].email), stdin);
    contacts[*count].email[strcspn(contacts[*count].email,
"\n")] = 0; // Remove newline
 (*count)++;
    printf("Contact added successfully.\n");
}
// Function to delete a contact by name
void deleteContact(Contact contacts[], int *count) {
    char name[50];
    int found = 0;
 if (*count == 0) {
        printf("No contacts to delete.\n");
        return;
```

```
    }
    printf("Enter the name of the contact to delete: ");
    getchar(); // Clear newline
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = 0; // Remove newline

    for (int i = 0; i < *count; i++) {
        if (strcmp(contacts[i].name, name) == 0) {
            for (int j = i; j < *count - 1; j++) {
                contacts[j] = contacts[j + 1];
            }
            (*count)--;
            printf("Contact deleted successfully.\n");
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Contact not found.\n");
    }
```

```
}
// Function to update contact information
void updateContact(Contact contacts[], int count) {
    char name[50];
    int found = 0;
  if (count == 0) {
        printf("No contacts to update.\n");
        return;
    }
 printf("Enter the name of the contact to update: ");
    getchar(); // Clear newline
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = 0; // Remove newline
  for (int i = 0; i < count; i++) {
        if (strcmp(contacts[i].name, name) == 0) {
            printf("Enter new phone number: ");
            fgets(contacts[i].phone, sizeof(contacts[i].phone),
stdin);
            contacts[i].phone[strcspn(contacts[i].phone, "\n")] = 0;
// Remove newline
```

```
            printf("Enter new email: ");
            fgets(contacts[i].email, sizeof(contacts[i].email), stdin);
            contacts[i].email[strcspn(contacts[i].email, "\n")] = 0;
// Remove newline
        printf("Contact updated successfully.\n");
            found = 1;
            break;
        }
    }
  if (!found) {
        printf("Contact not found.\n");
    }
}
// Function to search for a contact by name or phone number
void searchContact(Contact contacts[], int count) {
    char searchTerm[50];
    int found = 0;
      if (count == 0) {
        printf("No contacts available to search.\n");
        return;
```

```
    }
     printf("Enter name or phone number to search: ");
    getchar(); // Clear newline
    fgets(searchTerm, sizeof(searchTerm), stdin);
    searchTerm[strcspn(searchTerm, "\n")] = 0; // Remove
newline
     for (int i = 0; i < count; i++) {
        if (strstr(contacts[i].name, searchTerm) != NULL ||
strstr(contacts[i].phone, searchTerm) != NULL) {
            printf("\nContact Found:\n");
            printf("Name: %s\n", contacts[i].name);
            printf("Phone: %s\n", contacts[i].phone);
            printf("Email: %s\n", contacts[i].email);
            found = 1;
        }
    }
     if (!found) {
        printf("No contact found matching the search term.\n");}
}
// Function to display all contacts
void displayContacts(Contact contacts[], int count) {
```

```
    if (count == 0) {
        printf("No contacts to display.\n");
        return;
    }
  printf("\nList of Contacts:\n");
    for (int i = 0; i < count; i++) {
        printf("Name: %s\n", contacts[i].name);
        printf("Phone: %s\n", contacts[i].phone);
        printf("Email: %s\n", contacts[i].email);
        printf("-------------------------\n");
    }
}
```

Explanation of the Code:

1. **Contact Structure:**
    - **Each contact contains a name, phone number, and email address.**
2. **Menu System:**

- **The program provides options to add, delete, update, search, and display contacts. The user can select an option from the menu.**

3. Add Contact:

- **The user inputs the contact's name, phone number, and email. These details are added to the contact list.**

4. Delete Contact:

- **The user provides the name of the contact to be deleted. If found, the contact is removed, and the list is updated.**

5. Update Contact:

- **The user provides the name of the contact to be updated. If found, the user can modify the contact's phone number and email.**

6. Search Contact:

- **The user can search for a contact by either name or phone number. If found, the contact's details are displayed.**

7. Display Contacts:

- **All contacts in the address book are displayed.**

Sample Output:

Simple Address Book

1. Add Contact

2. Delete Contact

3. Update Contact

4. Search Contact

5. Display Contacts

6. Exit

Enter your choice: 1

Enter contact name: John Doe

Enter contact phone number: 123-456-7890

Enter contact email: john.doe@example.com

Contact added successfully.

Simple Address Book

1. Add Contact

2. Delete Contact

3. Update Contact

4. Search Contact

5. Display Contacts

6. Exit

Enter your choice: 5

List of Contacts:

Name: John Doe

Phone: 123-456-7890

Email: john.doe@example.com

Project Summary:

This Simple Address Book application allows users to manage contacts by adding, deleting, updating, and searching for contacts based on name or phone number. The program uses a basic menu-driven interface, making it user-friendly and easy to navigate.