Contact Management System using a Singly Linked List.

Chaitanya Prashant Patare

25092030

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Define the structure for a contact node

struct ContactNode {

   char name[50];

   char phone[15];

   char email[50];

   struct ContactNode *next; // Pointer to the next contact

};


// Global head pointer, initially NULL (empty list)

struct ContactNode *head = NULL;


// Function prototypes

void addContact();

void searchContact();

void deleteContact();

void displayAllContacts();


int main() {

   int choice;
```

```
while (1) {
    printf("\n--- Contact Management System ---\n");
    printf("1. Add New Contact\n");
    printf("2. Search Contact (by name)\n");
    printf("3. Delete Contact (by name)\n");
    printf("4. Display All Contacts\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    // Clear the input buffer
    while (getchar() != '\n');

    switch (choice) {
        case 1:
            addContact();
            break;
        case 2:
            searchContact();
            break;
        case 3:
            deleteContact();
            break;
        case 4:
            displayAllContacts();
            break;
        case 5:
            printf("Exiting program.\n");
            // Free all allocated memory before exiting
            while(head != NULL) {
                struct ContactNode *temp = head;
                head = head->next;
                free(temp);
            }
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

// Function to add a new contact to the end of the list
void addContact() {
    // 1. Allocate memory for the new node
    struct ContactNode *newNode = (struct ContactNode*) malloc(sizeof(struct
ContactNode));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }

    // 2. Get contact details from user
    printf("Enter Name: ");
```

```
fgets(newNode->name, 50, stdin);
newNode->name[strcspn(newNode->name, "\n")] = 0; // Remove newline character

printf("Enter Phone: ");
fgets(newNode->phone, 15, stdin);
newNode->phone[strcspn(newNode->phone, "\n")] = 0;

printf("Enter Email: ");
fgets(newNode->email, 50, stdin);
newNode->email[strcspn(newNode->email, "\n")] = 0;

// 3. Set the new node's next pointer
newNode->next = NULL;

// 4. Link the new node to the list
if (head == NULL) {
  // If the list is empty, make this the head
  head = newNode;
} else {
  // Find the last node and add the new node after it
  struct ContactNode *temp = head;
  while (temp->next != NULL) {
    temp = temp->next;
  }
  temp->next = newNode;
}

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

```
}

// Function to display all contacts
void displayAllContacts() {
    struct ContactNode *temp = head;

    if (temp == NULL) {
        printf("Contact list is empty.\n");
        return;
    }

    printf("\n--- All Contacts ---\n");
    int count = 1;
    while (temp != NULL) {
        printf("Contact #%d\n", count);
        printf("  Name:  %s\n", temp->name);
        printf("  Phone: %s\n", temp->phone);
        printf("  Email: %s\n", temp->email);
        printf("----------------------\n");
        temp = temp->next;
        count++;
    }
}

// Function to search for a contact by name
void searchContact() {
    char searchName[50];
    printf("Enter name to search: ");
    fgets(searchName, 50, stdin);
    searchName[strcspn(searchName, "\n")] = 0; // Remove newline

    struct ContactNode *temp = head;
    int found = 0;

    while (temp != NULL) {
        // Use strcmp for string comparison
        if (strcmp(temp->name, searchName) == 0) {
            printf("\n--- Contact Found ---\n");
            printf("  Name:  %s\n", temp->name);
            printf("  Phone: %s\n", temp->phone);
            printf("  Email: %s\n", temp->email);
            printf("---------------------\n");
            found = 1;
            break; // Found the contact, no need to search further
        }
        temp = temp->next;
    }

    if (found == 0) {
        printf("Contact with name '%s' not found.\n", searchName);
    }
}

// Function to delete a contact by name
void deleteContact() {
    char deleteName[50];
```

```
printf("Enter name to delete: ");
fgets(deleteName, 50, stdin);
deleteName[strcspn(deleteName, "\n")] = 0; // Remove newline

struct ContactNode *current = head;
struct ContactNode *previous = NULL;

// Check if list is empty
if (head == NULL) {
    printf("Contact list is empty.\n");
    return;
}

// Case 1: Delete the head node
if (strcmp(head->name, deleteName) == 0) {
    head = head->next; // Move head to the next node
    free(current);     // Free the old head
    printf("Contact '%s' deleted successfully.\n", deleteName);
    return;
}

// Case 2: Delete a node other than the head
// Traverse to find the node, keeping track of the previous node
while (current != NULL && strcmp(current->name, deleteName) != 0) {
    previous = current;
    current = current->next;
}
```

```
    // If the node was not found
    if (current == NULL) {
        printf("Contact with name '%s' not found.\n", deleteName);
        return;
    }

    // Node was found, so 'current' points to it and 'previous' points to the one before it
    previous->next = current->next; // Unlink the node
    free(current);                  // Free the memory
    printf("Contact '%s' deleted successfully.\n", deleteName);
}
```