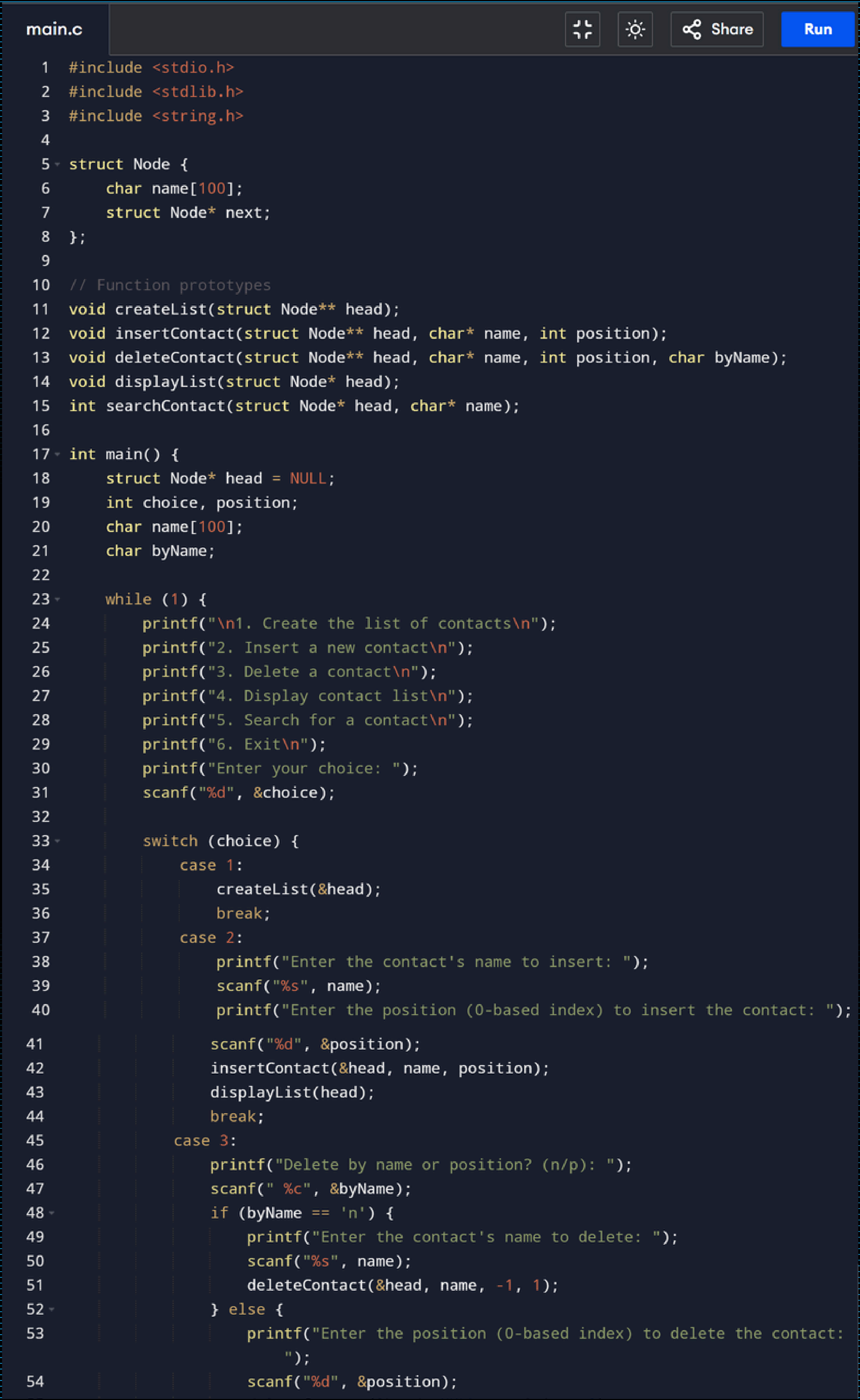
DSA ASSIGMENT-3

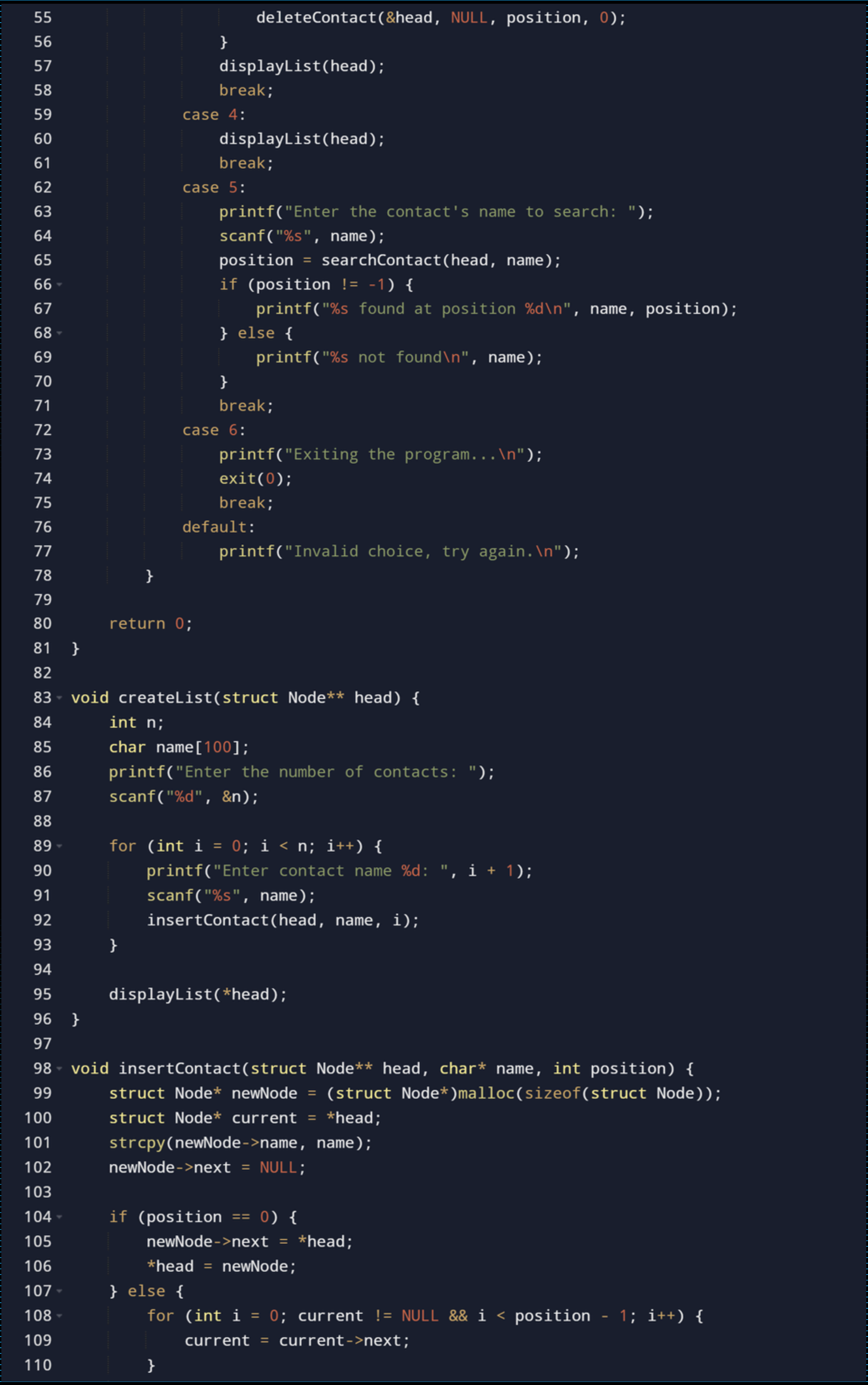
BY: ADHITHIYAN.B
RA2311026050030
GITHUB:HTTPS://GITHUB
COM/ADHI0007

1. You are given a task of implementing a simple contact management system using a singly linked list. The system will manage contact names. Implement the following operations using a singly linked list and switch case. After every operation, display the current list of contacts.

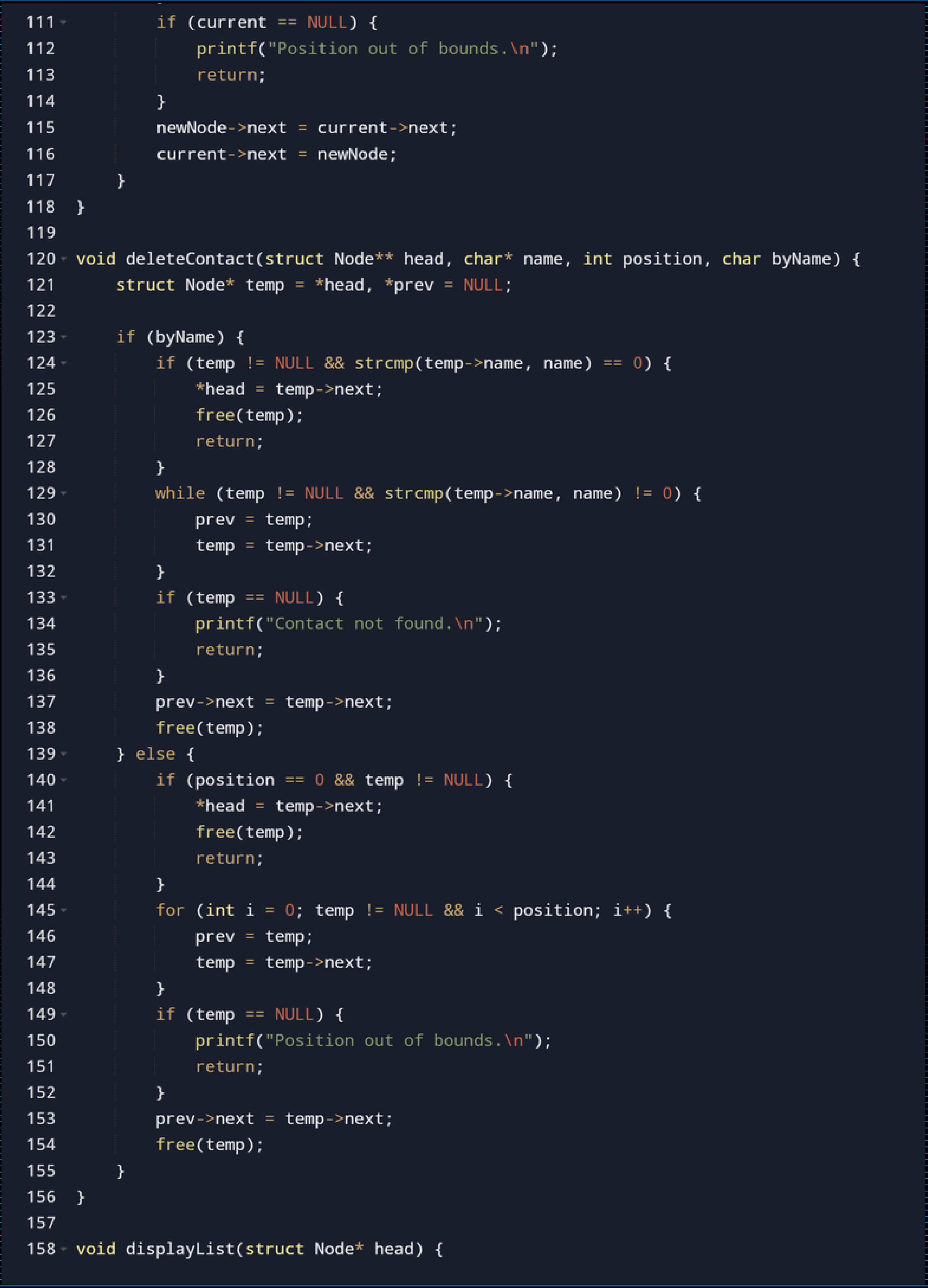
The operations to implement are:
(i) Creation of the list: Allow the user to create a list of contact names by enteringthem one by one.
(ii) Insertion of a new contact: Insert a new contact's name into a specific position in the list. The user should provide the name and the position at which it should be inserted.
(iii) Deletion of a contact: Delete a contact's name from the list based on their position or name. Ask the user whether they want to delete by name or by position.
(iv) Traversal of the list: Display all the contact names in the list in the current order.
(v) Search for a contact: Search for a contact's name in the list and display whether or not the contact is found, along with their position if present.



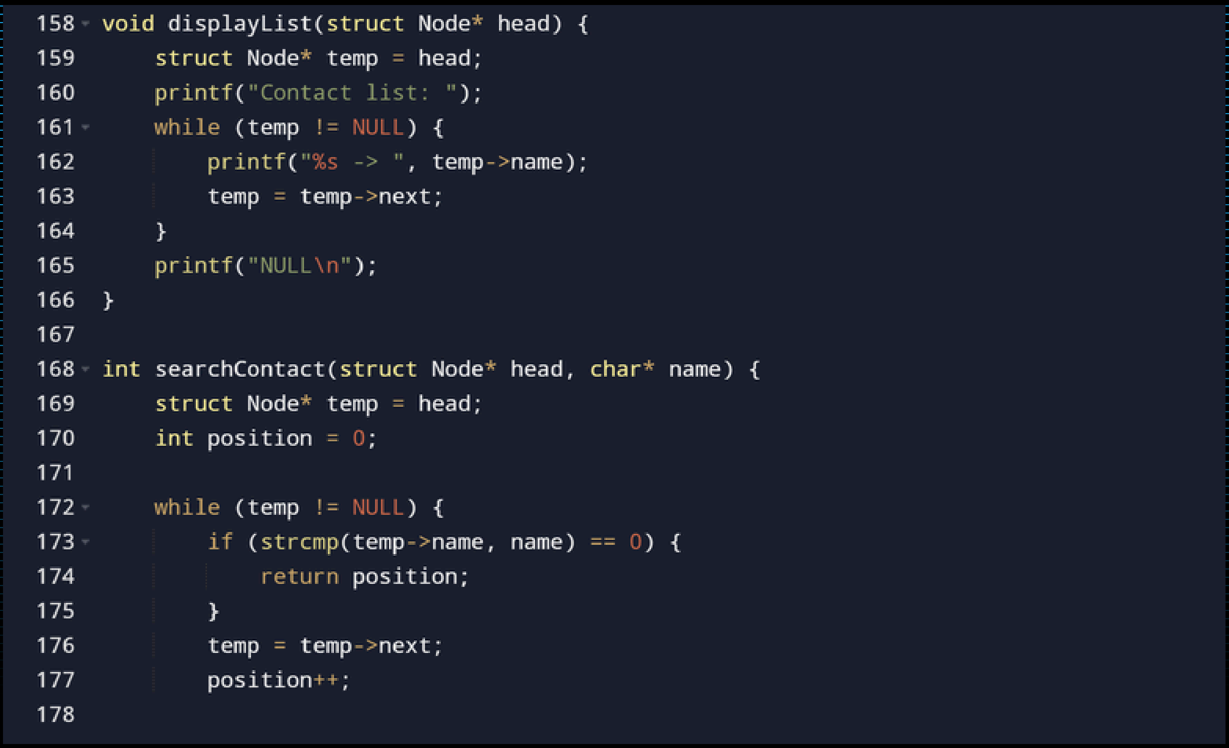
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Node {
    char name[100];
    struct Node* next;
};

// Function prototypes
void createList(struct Node** head);
void insertContact(struct Node** head, char* name, int position);
void deleteContact(struct Node** head, char* name, int position, char byName);
void displayList(struct Node* head);
int searchContact(struct Node* head, char* name);

int main() {
    struct Node* head = NULL;
    int choice, position;
    char name[100];
    char byName;

    while (1) {
        printf("\n1. Create the list of contacts\n");
        printf("2. Insert a new contact\n");
        printf("3. Delete a contact\n");
        printf("4. Display contact list\n");
        printf("5. Search for a contact\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createList(&head);
                break;
            case 2:
                printf("Enter the contact's name to insert: ");
                scanf("%s", name);
                printf("Enter the position (0-based index) to insert the contact: ");
                scanf("%d", &position);
                insertContact(&head, name, position);
                displayList(head);
                break;
            case 3:
                printf("Delete by name or position? (n/p): ");
                scanf(" %c", &byName);
                if (byName == 'n') {
                    printf("Enter the contact's name to delete: ");
                    scanf("%s", name);
                    deleteContact(&head, name, -1, 1);
                } else {
                    printf("Enter the position (0-based index) to delete the contact: ");
                    scanf("%d", &position);
```



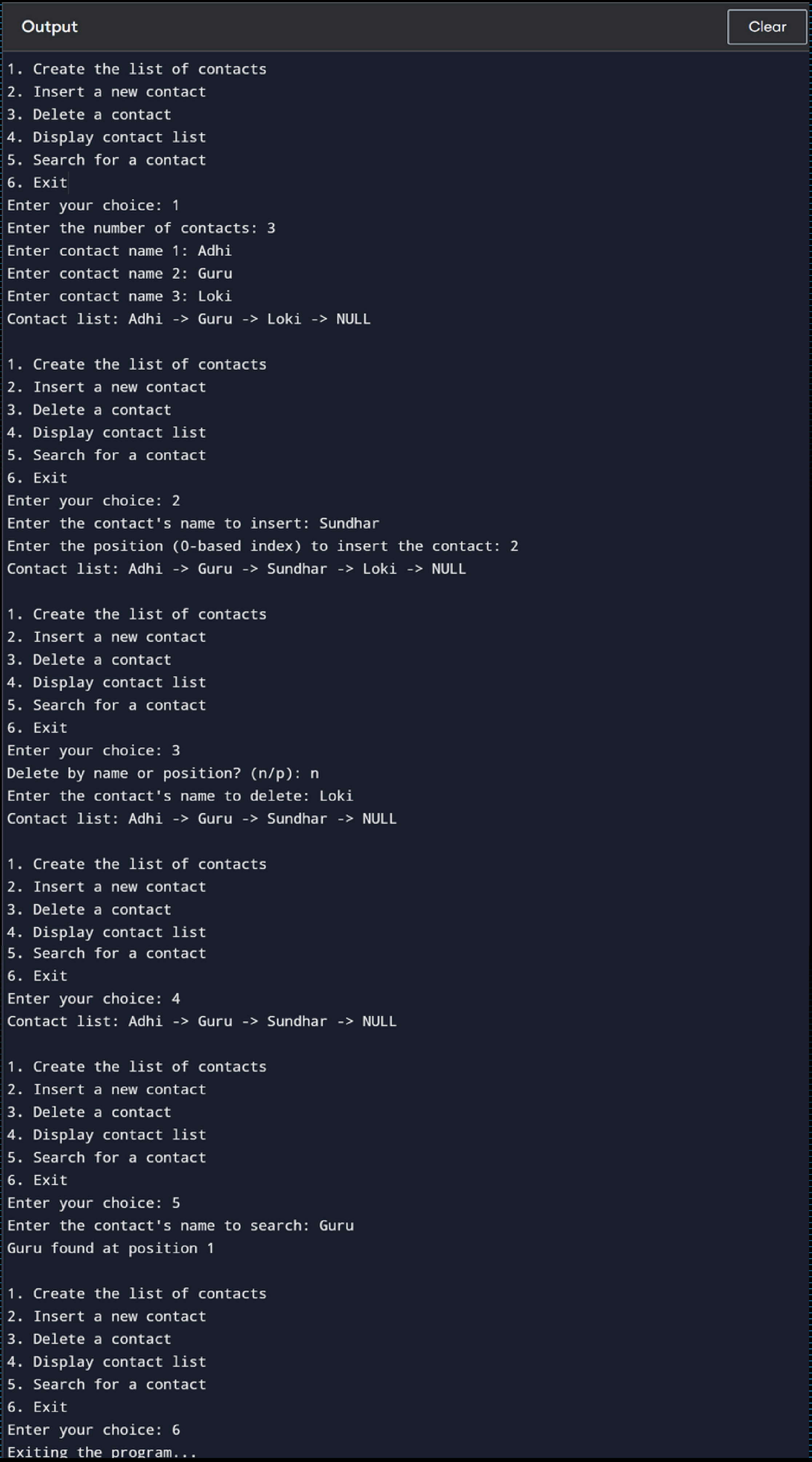
```c
                    deleteContact(&head, NULL, position, 0);
                }
                displayList(head);
                break;
            case 4:
                displayList(head);
                break;
            case 5:
                printf("Enter the contact's name to search: ");
                scanf("%s", name);
                position = searchContact(head, name);
                if (position != -1) {
                    printf("%s found at position %d\n", name, position);
                } else {
                    printf("%s not found\n", name);
                }
                break;
            case 6:
                printf("Exiting the program...\n");
                exit(0);
                break;
            default:
                printf("Invalid choice, try again.\n");
        }

    return 0;
}

void createList(struct Node** head) {
    int n;
    char name[100];
    printf("Enter the number of contacts: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter contact name %d: ", i + 1);
        scanf("%s", name);
        insertContact(head, name, i);
    }

    displayList(*head);
}

void insertContact(struct Node** head, char* name, int position) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    struct Node* current = *head;
    strcpy(newNode->name, name);
    newNode->next = NULL;

    if (position == 0) {
        newNode->next = *head;
        *head = newNode;
    } else {
        for (int i = 0; current != NULL && i < position - 1; i++) {
            current = current->next;
        }
```



```c
        if (current == NULL) {
            printf("Position out of bounds.\n");
            return;
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}

void deleteContact(struct Node** head, char* name, int position, char byName) {
    struct Node* temp = *head, *prev = NULL;

    if (byName) {
        if (temp != NULL && strcmp(temp->name, name) == 0) {
            *head = temp->next;
            free(temp);
            return;
        }
        while (temp != NULL && strcmp(temp->name, name) != 0) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Contact not found.\n");
            return;
        }
        prev->next = temp->next;
        free(temp);
    } else {
        if (position == 0 && temp != NULL) {
            *head = temp->next;
            free(temp);
            return;
        }
        for (int i = 0; temp != NULL && i < position; i++) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Position out of bounds.\n");
            return;
        }
        prev->next = temp->next;
        free(temp);
    }
}

void displayList(struct Node* head) {
```



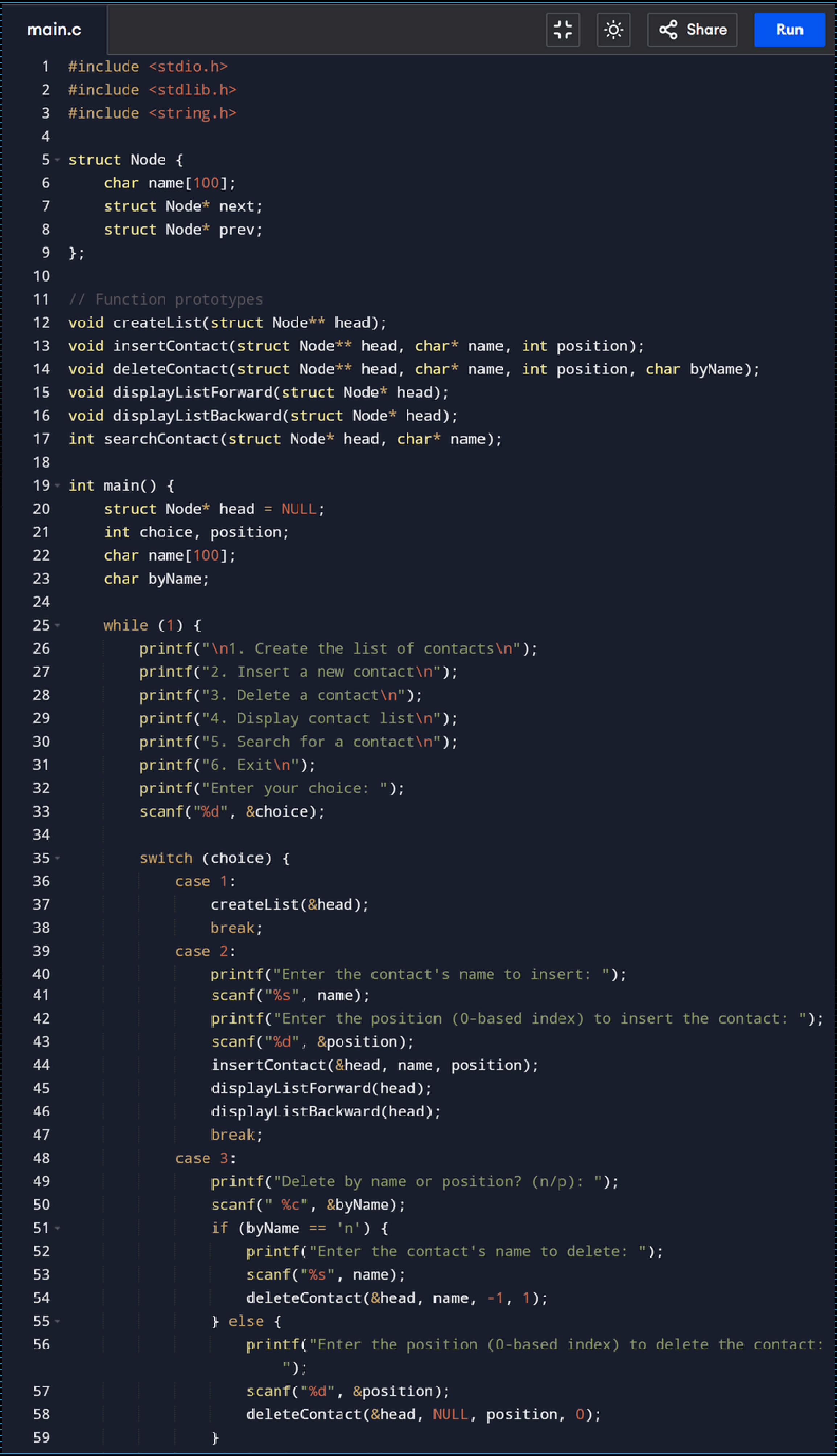
```c
void displayList(struct Node* head) {
    struct Node* temp = head;
    printf("Contact list: ");
    while (temp != NULL) {
        printf("%s -> ", temp->name);
        temp = temp->next;
    }
    printf("NULL\n");
}

int searchContact(struct Node* head, char* name) {
    struct Node* temp = head;
    int position = 0;

    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            return position;
        }
        temp = temp->next;
        position++;

```



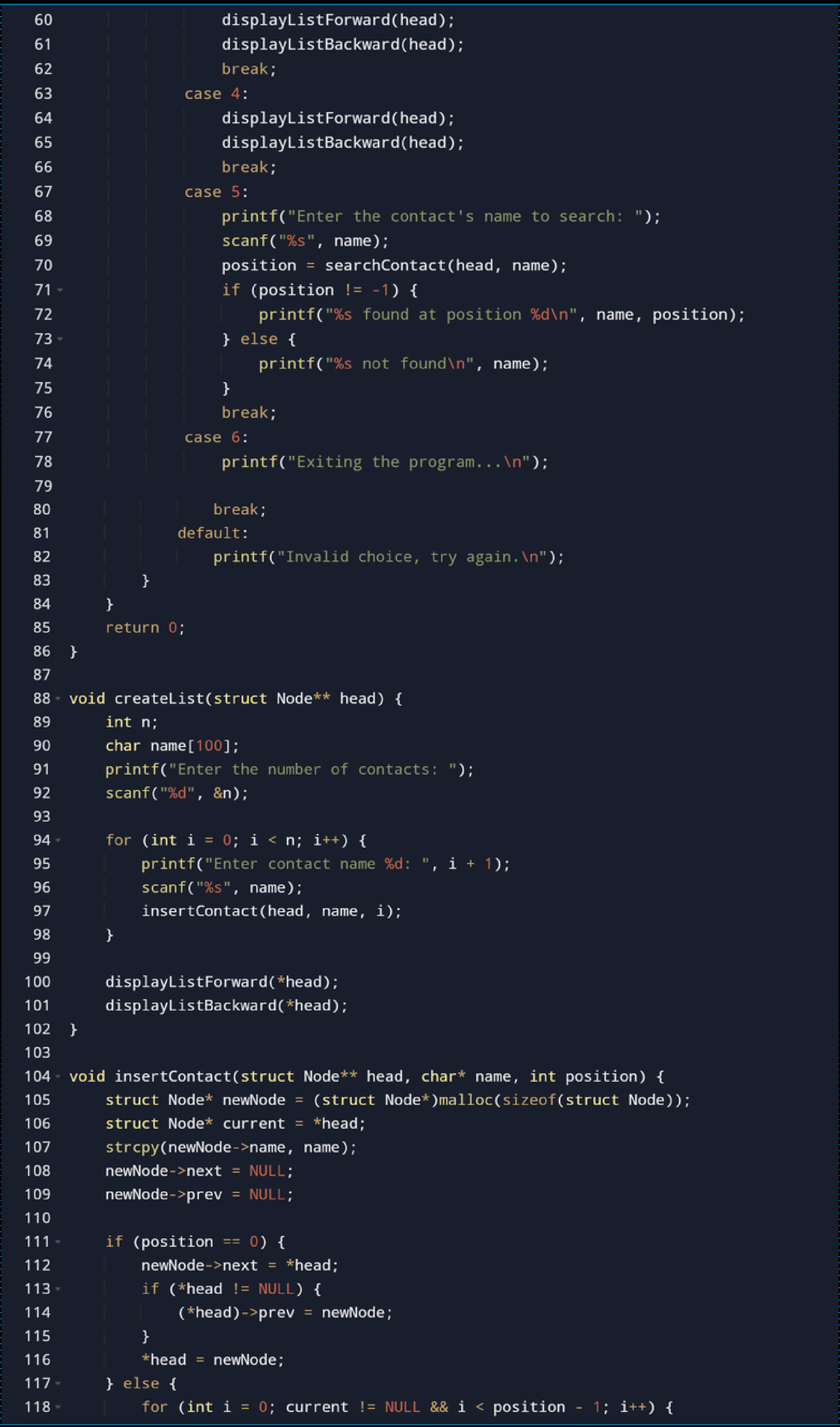
```
1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 1
Enter the number of contacts: 3
Enter contact name 1: Adhi
Enter contact name 2: Guru
Enter contact name 3: Loki
Contact list: Adhi -> Guru -> Loki -> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 2
Enter the contact's name to insert: Sundhar
Enter the position (0-based index) to insert the contact: 2
Contact list: Adhi -> Guru -> Sundhar -> Loki -> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 3
Delete by name or position? (n/p): n
Enter the contact's name to delete: Loki
Contact list: Adhi -> Guru -> Sundhar -> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 4
Contact list: Adhi -> Guru -> Sundhar -> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 5
Enter the contact's name to search: Guru
Guru found at position 1

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 6
Exiting the program...
```

2. You are tasked with implementing a simple contact management system using a doubly linked list. The system will manage contact names. Implement the following operations using a doubly linked list and switch-case. After every operation, display the current list of contacts.

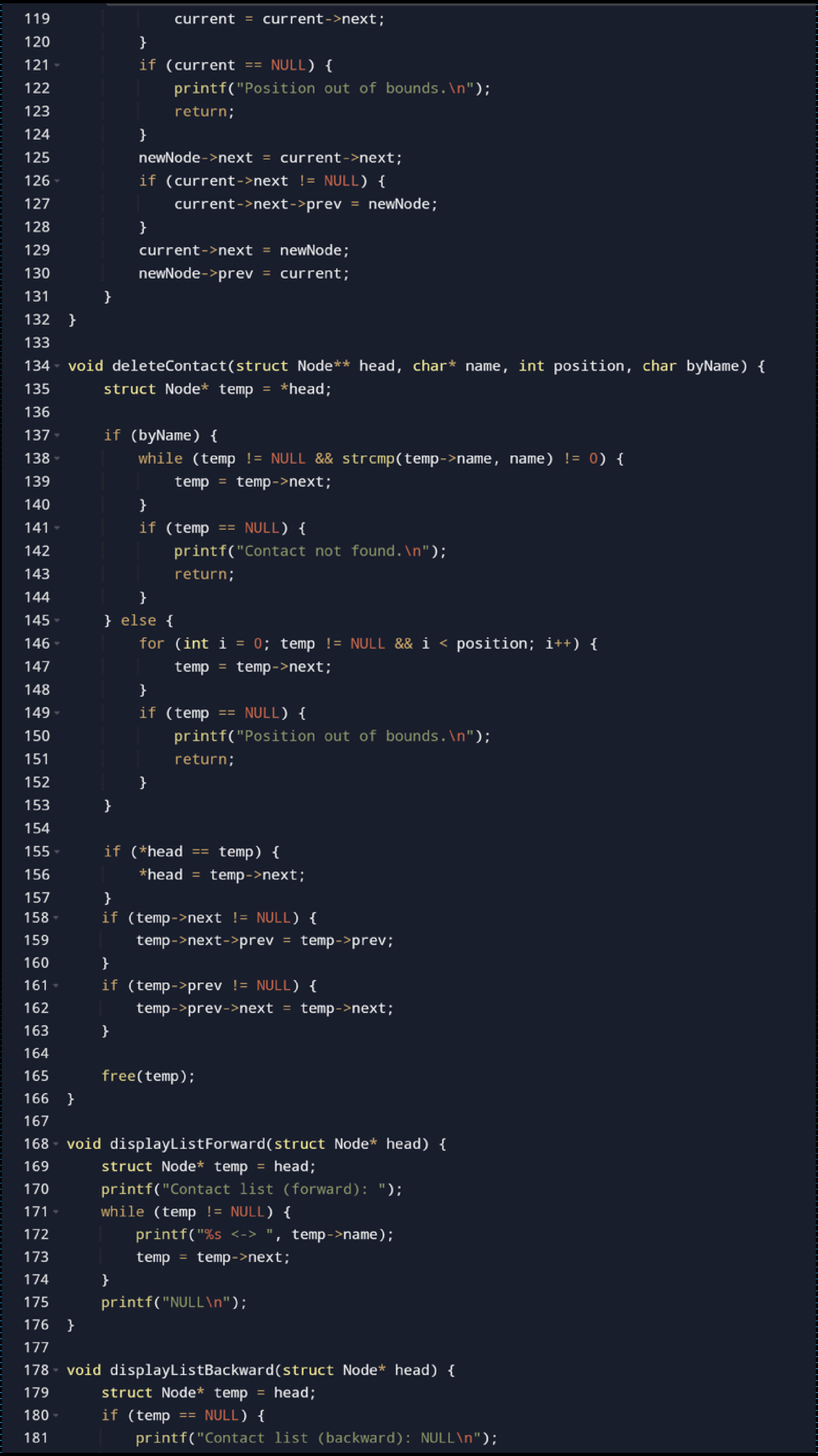
The operations to implement are:
***(i) Creation of the list:** Allow the user to create a list of contact names by entering them one by one.*
***(ii) Insertion of a new contact:** Insert a new contact's name into a specific position in the list. The user should provide the name and the position at which it should be inserted.*
***(iii)Deletion of a contact:** Delete a contact's name from the list based on their position or name. Ask the user whether they want to delete by name or by position.*
***(iv)Traversal of the list (in both directions):** Display all the contact names in the list in the current order (forward traversal) and then display them in reverse order (backward traversal).*
***(v) Search for a contact:** Search for a contact's name in the list and display whether or not the contact is found, along with their position if present.*



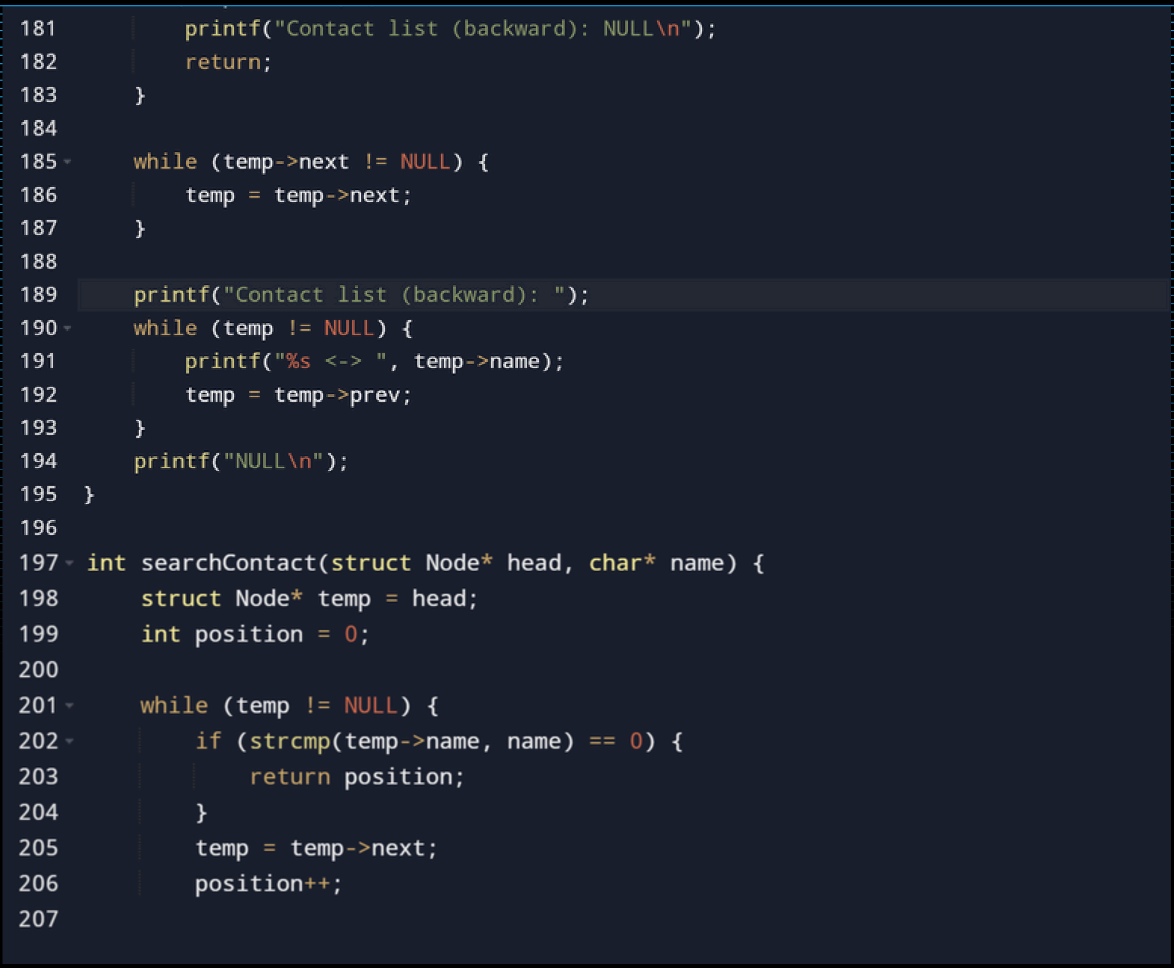
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Node {
    char name[100];
    struct Node* next;
    struct Node* prev;
};

// Function prototypes
void createList(struct Node** head);
void insertContact(struct Node** head, char* name, int position);
void deleteContact(struct Node** head, char* name, int position, char byName);
void displayListForward(struct Node* head);
void displayListBackward(struct Node* head);
int searchContact(struct Node* head, char* name);

int main() {
    struct Node* head = NULL;
    int choice, position;
    char name[100];
    char byName;

    while (1) {
        printf("\n1. Create the list of contacts\n");
        printf("2. Insert a new contact\n");
        printf("3. Delete a contact\n");
        printf("4. Display contact list\n");
        printf("5. Search for a contact\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createList(&head);
                break;
            case 2:
                printf("Enter the contact's name to insert: ");
                scanf("%s", name);
                printf("Enter the position (0-based index) to insert the contact: ");
                scanf("%d", &position);
                insertContact(&head, name, position);
                displayListForward(head);
                displayListBackward(head);
                break;
            case 3:
                printf("Delete by name or position? (n/p): ");
                scanf(" %c", &byName);
                if (byName == 'n') {
                    printf("Enter the contact's name to delete: ");
                    scanf("%s", name);
                    deleteContact(&head, name, -1, 1);
                } else {
                    printf("Enter the position (0-based index) to delete the contact:
                        ");
                    scanf("%d", &position);
                    deleteContact(&head, NULL, position, 0);
                }
```



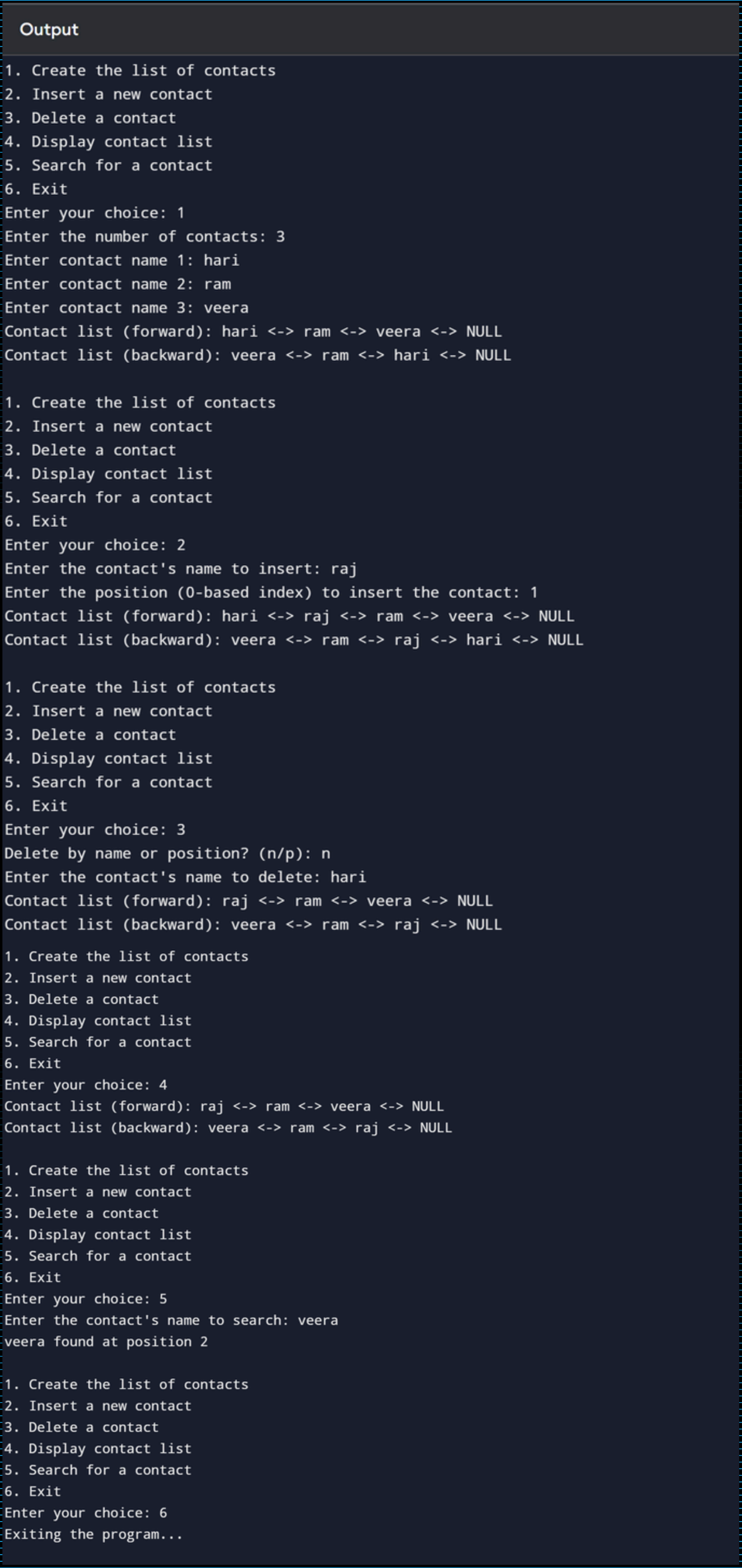
```c
                displayListForward(head);
                displayListBackward(head);
                break;
            case 4:
                displayListForward(head);
                displayListBackward(head);
                break;
            case 5:
                printf("Enter the contact's name to search: ");
                scanf("%s", name);
                position = searchContact(head, name);
                if (position != -1) {
                    printf("%s found at position %d\n", name, position);
                } else {
                    printf("%s not found\n", name);
                }
                break;
            case 6:
                printf("Exiting the program...\n");

                break;
            default:
                printf("Invalid choice, try again.\n");
        }
    }
    return 0;
}

void createList(struct Node** head) {
    int n;
    char name[100];
    printf("Enter the number of contacts: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter contact name %d: ", i + 1);
        scanf("%s", name);
        insertContact(head, name, i);
    }

    displayListForward(*head);
    displayListBackward(*head);
}

void insertContact(struct Node** head, char* name, int position) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    struct Node* current = *head;
    strcpy(newNode->name, name);
    newNode->next = NULL;
    newNode->prev = NULL;

    if (position == 0) {
        newNode->next = *head;
        if (*head != NULL) {
            (*head)->prev = newNode;
        }
        *head = newNode;
    } else {
        for (int i = 0; current != NULL && i < position - 1; i++) {
```



```c
            current = current->next;
        }
        if (current == NULL) {
            printf("Position out of bounds.\n");
            return;
        }
        newNode->next = current->next;
        if (current->next != NULL) {
            current->next->prev = newNode;
        }
        current->next = newNode;
        newNode->prev = current;
    }
}

void deleteContact(struct Node** head, char* name, int position, char byName) {
    struct Node* temp = *head;

    if (byName) {
        while (temp != NULL && strcmp(temp->name, name) != 0) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Contact not found.\n");
            return;
        }
    } else {
        for (int i = 0; temp != NULL && i < position; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Position out of bounds.\n");
            return;
        }
    }

    if (*head == temp) {
        *head = temp->next;
    }
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }

    free(temp);
}

void displayListForward(struct Node* head) {
    struct Node* temp = head;
    printf("Contact list (forward): ");
    while (temp != NULL) {
        printf("%s <-> ", temp->name);
        temp = temp->next;
    }
    printf("NULL\n");
}

void displayListBackward(struct Node* head) {
    struct Node* temp = head;
    if (temp == NULL) {
        printf("Contact list (backward): NULL\n");
```



```c
        printf("Contact list (backward): NULL\n");
        return;
    }

    while (temp->next != NULL) {
        temp = temp->next;
    }

    printf("Contact list (backward): ");
    while (temp != NULL) {
        printf("%s <-> ", temp->name);
        temp = temp->prev;
    }
    printf("NULL\n");
}

int searchContact(struct Node* head, char* name) {
    struct Node* temp = head;
    int position = 0;

    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            return position;
        }
        temp = temp->next;
        position++;

```


Output

```
1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 1
Enter the number of contacts: 3
Enter contact name 1: hari
Enter contact name 2: ram
Enter contact name 3: veera
Contact list (forward): hari <-> ram <-> veera <-> NULL
Contact list (backward): veera <-> ram <-> hari <-> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 2
Enter the contact's name to insert: raj
Enter the position (0-based index) to insert the contact: 1
Contact list (forward): hari <-> raj <-> ram <-> veera <-> NULL
Contact list (backward): veera <-> ram <-> raj <-> hari <-> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 3
Delete by name or position? (n/p): n
Enter the contact's name to delete: hari
Contact list (forward): raj <-> ram <-> veera <-> NULL
Contact list (backward): veera <-> ram <-> raj <-> NULL
1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 4
Contact list (forward): raj <-> ram <-> veera <-> NULL
Contact list (backward): veera <-> ram <-> raj <-> NULL

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 5
Enter the contact's name to search: veera
veera found at position 2

1. Create the list of contacts
2. Insert a new contact
3. Delete a contact
4. Display contact list
5. Search for a contact
6. Exit
Enter your choice: 6
Exiting the program...
```