Madhav K Mohan
RA2311026050003
AIML B

DSA ASSIGNMENT 3

Singly and Doubly Linked List

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 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: 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.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Node {
    char name[50];
    struct Node* next;
};
struct Node* createNode(char* name) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    strcpy(newNode->name, name);
    newNode->next = NULL;
    return newNode;
}
void displayList(struct Node* head) {
    struct Node* temp = head;
    if (temp == NULL) {
        printf("Contact list is empty. \n");
        return;
    }
    printf("Contact list: ");
    while (temp != NULL) {
        printf ("%s -> ", temp->name);
        temp = temp->next;
    }
    printf("NULL\n");
}
struct Node* createList() {
    int n;
    char name[50];
    struct Node* head = NULL;
    struct Node* temp = NULL;
    printf("Enter the number of contacts: ");
    scanf ("%d", &n);
    getchar();
    for (int i = 0; i < n; i++) {
        printf("Enter contact name %d: ", i + 1);
        fgets(name, 50, stdin);
        name[strcspn(name, "\n")] = 0;
        struct Node* newNode = createNode (name);
        if (head == NULL) {
            head = newNode;
        } else {
            temp->next = newNode;
```

```
        }
        temp = newNode;
    }
    displayList(head);
    return head;
}
struct Node* insertContact(struct Node* head) {
    char name[50];
    int pos, i = 0;
    printf("Enter the contact's name to insert: ");
    getchar();
    fgets (name, 50, stdin);
    name [strcspn(name, "\n")] = 0;
    printf("Enter the position (0-based index) to insert the contact: ");
    scanf ("%d", &pos);
    struct Node* newNode = createNode(name);
    if (pos == 0) {
        newNode->next = head;
        head = newNode;
    } else {
        struct Node* temp = head;
        while (i < pos - 1 && temp != NULL) {
            temp = temp->next;
            i++;
        }
        if (temp == NULL) {
        displayList(head);
        return head;
    }
}
struct Node* deleteContact(struct Node* head) {
    if (head == NULL) {
        printf("The contact list is empty. \n");
        return head;
    }
    char choice, name[50];
    int pos;
    printf("Delete by name or position? (n/p): ");
    getchar();
    scanf("%c", &choice);
    if (choice == 'n') {
        printf("Enter the contact's name to delete: ");
```

```
        getchar();
        fgets(name, 50, stdin);
        name[strcspn(name, "\n")] = 0;
        struct Node* temp = head;
        struct Node* prev = NULL;
        while (temp != NULL && strcmp(temp->name, name) != 0) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Contact not found. \n");
            return head;
        }
        if (prev == NULL) {
            head = temp->next;
        } else {
            prev->next = temp->next;
        }
        free(temp);
    } else if (choice == 'p') {
        printf("Enter the position to delete the contact: ");
        scanf ("%d", &pos);
        if (pos ==0) {
            struct Node* temp = head;
            head = head->next;
            free(temp);
        } else {
            struct Node* temp = head;
            struct Node* prev = NULL;
            int i = 0;
            while (i < pos && temp != NULL) {
                prev = temp;
                temp = temp->next;
                i++;
            }
            if (temp == NULL) {
                printf("Invalid position. \n");
                return head;
            }
            prev->next = temp->next;
            free(temp);
        }
    } else {
```

```
        printf("Invalid choice. \n");
    }
    displayList(head);
    return head;
}
void searchContact(struct Node* head) {
    char name[50];
    int pos = 0;
    struct Node* temp = head;
    printf("Enter the contact's name to search: ");
    getchar();
    fgets (name, 50, stdin);
    name [strcspn(name, "\n")] = 0;
    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            printf("%s found at position %d\n", name, pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("%s not found in the list. \n", name);
}
int main() {
    struct Node* head = NULL;
    int choice;
    do {
        printf("\n1. Create the list of contacts\n");
        printf("2. Insert a new contact\n");
        printf("3. Delete a contact\n");
        printf("4. Search for e contact\n");
        printf("6. Exit\n");
        printf( "Enter your choice: ");
        scanf ("%d", &choice);
        switch (choice) {
            case 1:
                head = createList();
                break;
            case 2:
                head = insertContact(head);
                break;
           case 3:
               head = deleteContact(head);
               break;
           case 4:
               displayList(head);
               break;
           case 5:
               searchContact(head);
               break;
           case 6:
               printf("Exiting the program... \n");
               break;
           default:
               printf("Invalid choice. Please try again.\n");
       }
    } while (choice != 6);
    return 0;
}
```

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:
```

```
Enter your choice: 1
Enter the number of contacts: 3
Enter contact name 1: Madhav
Enter contact name 2: Aswin
Enter contact name 3: Manu
Contact list: Madhav -> Aswin -> Manu -> NULL
```

```
Enter your choice: 2
Enter the contact's name to insert: Naveen
Enter the position (0-based index) to insert the contact: 3
Contact list: Madhav -> Aswin -> Manu -> Naveen -> NULL
```

```
Enter your choice: 3
Delete by name or position? (n/p): p
Enter the position to delete the contact: 2
Contact list: Madhav -> Aswin -> Naveen -> NULL
```

```
Enter your choice: 4
Contact list: Madhav -> Aswin -> Naveen -> NULL
```

```
Enter your choice: 5
Enter the contact's name to search: Aswin
Aswin found at position 1
```

```
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[50];
    struct Node* prev;
    struct Node* next;
};
int main() {
    struct Node *head = NULL, *tail = NULL, *temp, *newNode;
    int choice, pos, size = 0, i;
    char name[50], delchoice;
    do {
        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:
                printf("Enter the number of contacts: ");
                scanf ("%d", &size);
                getchar();
                for (i = 0; i < size; i++) {
                    printf("Enter contact name %d: ", i + 1);
                    fgets (name, 50, stdin);
                    name[strcspn(name, "\n")] = 0;
                    newNode = (struct Node*)malloc(sizeof(struct Node));
                    strcpy(newNode->name, name);
                    newNode->prev = NULL;
                    newNode->next = NULL;
                    if (head == NULL) {
                        head = newNode;
                        tail = newNode;
                    } else {
                        tail->next = newNode;
                        newNode->prev = tail;
                        tail = newNode;
                    }
                }
```

```
            break;
        case 2:
            printf("Enter the contact's name to insert: ");
            getchar();
            fgets(name, 50, stdin);
            name[strcspn(name, "\n")] = 0;
            printf("Enter the position (0-based index) to insert the contact: ");
            scanf ("%d", &pos);
            if (pos < 0 || pos > size) {
                printf ("Invalid position. \n");
                break;
            }
            newNode = (struct Node*)malloc(sizeof(struct Node));
            strcpy(newNode->name, name);
            newNode->prev = NULL;
            newNode->next = NULL;
            if (pos == 0) {
                newNode->next = head;
                if (head != NULL) {
                    head->prev = newNode; }
                head = newNode;
            } else {
                temp = head;
                for (i = 0; i < pos - 1; i++) {
                    temp = temp->next;
                }
                newNode->next = temp->next;
                if (temp->next != NULL) {
                    temp->next->prev = newNode;
                }
                temp->next = newNode;
                newNode->prev = temp;
            }
            size++;
            break;
        case 3:
            printf("Delete by name or position? (n/p): ");
            getchar();
            scanf ("%c", &delchoice);
            if (delchoice == 'n') {
                printf("Enter the Contact's nane to delete: ");
                getchar();
```

```
            fgets(name, 50, stdin);
            name[strcspn(name, "\n")] = 0;
            temp = head;
            while (temp != NULL && strcmp(temp->name, name) != 0) {
                temp = temp->next;
            }
            if (temp == NULL) {
                printf("Contact not found. \n");
                break;
            }
        } else if (delchoice == 'p') {
            printf("Enter the position (0-based index) to delete the contact: ");
            scanf ("%d", &pos);
            if (pos < 0 || pos >= size) {
                printf("Invalid position. \n");
                break;
            }
            temp = head;
            for (i = 0; i < pos; i++) {
                temp = temp->next;
            }
        } else {
            printf("Invalid choice. \n");
            break;
        }
        if (temp->prev != NULL) {
            temp->prev->next = temp->next;
        } else {
            head = temp->next;
        }
        if (temp->next != NULL) {
            temp->next->prev = temp->prev;
        }
        free(temp);
        size--;
        break;
    case 4:
        temp = head;
        printf ("Contact list (forward): ");
        while (temp != NULL) {
            printf ("%s <->", temp->name);
            tail = temp;
            temp = temp->next;
```

```
129             }
130             printf ("NULL\n");
131             temp = tail;
132             printf("Contact list (backward): ");
133             while (temp != NULL) {
134                 printf("%s <->", temp->name);
135                 temp = temp->prev;
136             }
137             printf("NULL\n");
138             break;
139         case 5:
140             printf("Enter the contact's name to search: ");
141             getchar();
142             fgets(name, 50, stdin);
143             name[strcspn(name, "\n")] = 0;
144             temp = head;
145             i = 0;
146             while (temp != NULL) {
147                 if (strcmp(temp->name, name) == 0) {
148                     printf("%s found at position %d\n", name, i);
149                     break;
150                 }
151                 temp = temp->next;
152                 i++;
153             }
154             if (temp == NULL) {
155                 printf("%s not found in the list. \n", name);
156             }
157             break;
158         case 6:
159             printf("Exiting the program... \n");
160             break;
161         default:
162             printf("Invalid choice. Please try again. \n");
163     }
164 } while (choice !=6);
165 return 0;
166 }
```

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: Madhav
Enter contact name 2: Aswin
Enter contact name 3: Naveen

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: Sujin
Enter the position (0-based index) to insert the contact: 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: 3
Delete by name or position? (n/p): p
Enter the position (0-based index) to delete the contact: 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: 4
Contact list (forward): Madhav <->Sujin <->Naveen <->NULL
Contact list (backward): Naveen <->Sujin <->Madhav <->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: Madhav
Madhav found at position 0

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...


=== Code Execution Successful ===
```