

DSA ASSIGNMENT 3

Madhav K Mohan
RA2311026050003
AIML B

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.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  struct Node {
5      char name[50];
6      struct Node* next;
7  };
8  struct Node* createNode(char* name) {
9      struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
10     strcpy(newNode->name, name);
11     newNode->next = NULL;
12     return newNode;
13 }
14 void displayList(struct Node* head) {
15     struct Node* temp = head;
16     if (temp == NULL) {
17         printf("Contact list is empty. \n");
18         return;
19     }
20     printf("Contact list: ");
21     while (temp != NULL) {
22         printf ("%s -> ", temp->name);
23         temp = temp->next;
24     }
25     printf("NULL\n");
26 }
27 struct Node* createList() {
28     int n;
29     char name[50];
30     struct Node* head = NULL;
31     struct Node* temp = NULL;
32     printf("Enter the number of contacts: ");
33     scanf ("%d", &n);
34     getchar();
35     for (int i = 0; i < n; i++) {
36         printf("Enter contact name %d: ", i + 1);
37         fgets(name, 50, stdin);
38         name[strcspn(name, "\n")] = 0;
39         struct Node* newNode = createNode (name);
40         if (head == NULL) {
41             head = newNode;
42         } else {
43             temp->next = newNode;
```

```

44     }
45     temp = newNode;
46 }
47 displayList(head);
48 return head;
49 }
50 ▾ struct Node* insertContact(struct Node* head) {
51     char name[50];
52     int pos, i = 0;
53     printf("Enter the contact's name to insert: ");
54     getchar();
55     fgets (name, 50, stdin);
56     name [strcspn(name, "\n")] = 0;
57     printf("Enter the position (0-based index) to insert the contact: ");
58     scanf ("%d", &pos);
59     struct Node* newNode = createNode(name);
60 ▾ if (pos == 0) {
61         newNode->next = head;
62         head = newNode;
63 ▾ } else {
64         struct Node* temp = head;
65 ▾         while (i < pos - 1 && temp != NULL) {
66             temp = temp->next;
67             i++;
68         }
69 ▾         if (temp == NULL) {
70             displayList(head);
71             return head;
72         }
73 }
74 ▾ struct Node* deleteContact(struct Node* head) {
75 ▾     if (head == NULL) {
76         printf("The contact list is empty. \n");
77         return head;
78     }
79     char choice, name[50];
80     int pos;
81     printf("Delete by name or position? (n/p): ");
82     getchar();
83     scanf("%c", &choice);
84 ▾     if (choice == 'n') {
85         printf("Enter the contact's name to delete: ");

```

```

86     getchar();
87     fgets(name, 50, stdin);
88     name[strcspn(name, "\n")] = 0;
89     struct Node* temp = head;
90     struct Node* prev = NULL;
91     while (temp != NULL && strcmp(temp->name, name) != 0) {
92         prev = temp;
93         temp = temp->next;
94     }
95     if (temp == NULL) {
96         printf("Contact not found. \n");
97         return head;
98     }
99     if (prev == NULL) {
100         head = temp->next;
101     } else {
102         prev->next = temp->next;
103     }
104     free(temp);
105 } else if (choice == 'p') {
106     printf("Enter the position to delete the contact: ");
107     scanf ("%d", &pos);
108     if (pos ==0) {
109         struct Node* temp = head;
110         head = head->next;
111         free(temp);
112     } else {
113         struct Node* temp = head;
114         struct Node* prev = NULL;
115         int i = 0;
116         while (i < pos && temp != NULL) {
117             prev = temp;
118             temp = temp->next;
119             i++;
120         }
121         if (temp == NULL) {
122             printf("Invalid position. \n");
123             return head;
124         }
125         prev->next = temp->next;
126         free(temp);
127     }
128 } else {

```

```

129     printf("Invalid choice. \n");
130 }
131 displayList(head);
132 return head;
133 }
134 void searchContact(struct Node* head) {
135     char name[50];
136     int pos = 0;
137     struct Node* temp = head;
138     printf("Enter the contact's name to search: ");
139     getchar();
140     fgets (name, 50, stdin);
141     name [strcspn(name, "\n")] = 0;
142     while (temp != NULL) {
143         if (strcmp(temp->name, name) == 0) {
144             printf("%s found at position %d\n", name, pos);
145             return;
146         }
147         temp = temp->next;
148         pos++;
149     }
150     printf("%s not found in the list. \n", name);
151 }
152 int main() {
153     struct Node* head = NULL;
154     int choice;
155     do {
156         printf("\n1. Create the list of contacts\n");
157         printf("2. Insert a new contact\n");
158         printf("3. Delete a contact\n");
159         printf("4. Search for a contact\n");
160         printf("6. Exit\n");
161         printf("Enter your choice: ");
162         scanf ("%d", &choice);
163         switch (choice) {
164             case 1:
165                 head = createList();
166                 break;
167             case 2:
168                 head = insertContact(head);
169                 break;
170             case 3:
171                 head = deleteContact(head);
172                 break;
173             case 4:
174                 displayList(head);
175                 break;
176             case 5:
177                 searchContact(head);
178                 break;
179             case 6:
180                 printf("Exiting the program... \n");
181                 break;
182             default:
183                 printf("Invalid choice. Please try again.\n");
184         }
185     } while (choice != 6);
186     return 0;
187 }

```

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.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  struct Node {
5      char name[50];
6      struct Node* prev;
7      struct Node* next;
8  };
9  int main() {
10     struct Node *head = NULL, *tail = NULL, *temp, *newNode;
11     int choice, pos, size = 0, i;
12     char name[50], delchoice;
13     do {
14         printf("\n1. Create the list of contacts\n");
15         printf("2. Insert a new contact\n");
16         printf("3. Delete a contact\n");
17         printf("4. Display contact list\n");
18         printf("5. Search for a contact\n");
19         printf("6. Exit\n");
20         printf("Enter your choice: ");
21         scanf ("%d", &choice);
22         switch (choice) {
23             case 1:
24                 printf("Enter the number of contacts: ");
25                 scanf ("%d", &size);
26                 getchar();
27                 for (i = 0; i < size; i++) {
28                     printf("Enter contact name %d: ", i + 1);
29                     fgets (name, 50, stdin);
30                     name[strcspn(name, "\n")] = 0;
31                     newNode = (struct Node*)malloc(sizeof(struct Node));
32                     strcpy(newNode->name, name);
33                     newNode->prev = NULL;
34                     newNode->next = NULL;
35                     if (head == NULL) {
36                         head = newNode;
37                         tail = newNode;
38                     } else {
39                         tail->next = newNode;
40                         newNode->prev = tail;
41                         tail = newNode;
42                     }
43                 }
44             }
45     } while (choice != 6);
46 }
```

```

44         break;
45     case 2:
46         printf("Enter the contact's name to insert: ");
47         getchar();
48         fgets(name, 50, stdin);
49         name[strcspn(name, "\n")] = 0;
50         printf("Enter the position (0-based index) to insert the contact: ");
51         scanf ("%d", &pos);
52         if (pos < 0 || pos > size) {
53             printf ("Invalid position. \n");
54             break;
55         }
56         newNode = (struct Node*)malloc(sizeof(struct Node));
57         strcpy(newNode->name, name);
58         newNode->prev = NULL;
59         newNode->next = NULL;
60         if (pos == 0) {
61             newNode->next = head;
62             if (head != NULL) {
63                 head->prev = newNode; }
64             head = newNode;
65         } else {
66             temp = head;
67             for (i = 0; i < pos - 1; i++) {
68                 temp = temp->next;
69             }
70             newNode->next = temp->next;
71             if (temp->next != NULL) {
72                 temp->next->prev = newNode;
73             }
74             temp->next = newNode;
75             newNode->prev = temp;
76         }
77         size++;
78         break;
79     case 3:
80         printf("Delete by name or position? (n/p): ");
81         getchar();
82         scanf ("%c", &delchoice);
83         if (delchoice == 'n') {
84             printf("Enter the Contact's name to delete: ");
85             getchar();

```

```

86         fgets(name, 50, stdin);
87         name[strcspn(name, "\n")] = 0;
88         temp = head;
89         while (temp != NULL && strcmp(temp->name, name) != 0) {
90             temp = temp->next;
91         }
92         if (temp == NULL) {
93             printf("Contact not found. \n");
94             break;
95         }
96         } else if (delchoice == 'p') {
97             printf("Enter the position (0-based index) to delete the contact: ");
98             scanf ("%d", &pos);
99             if (pos < 0 || pos >= size) {
100                 printf("Invalid position. \n");
101                 break;
102             }
103             temp = head;
104             for (i = 0; i < pos; i++) {
105                 temp = temp->next;
106             }
107         } else {
108             printf("Invalid choice.\n");
109             break;
110         }
111         if (temp->prev != NULL) {
112             temp->prev->next = temp->next;
113         } else {
114             head = temp->next;
115         }
116         if (temp->next != NULL) {
117             temp->next->prev = temp->prev;
118         }
119         free(temp);
120         size--;
121         break;
122     case 4:
123         temp = head;
124         printf ("Contact list (forward): ");
125         while (temp != NULL) {
126             printf ("%s <->", temp->name);
127             tail = temp;
128             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 ===
```