

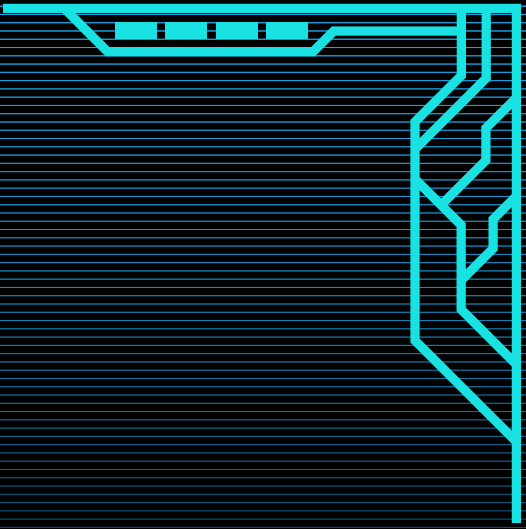
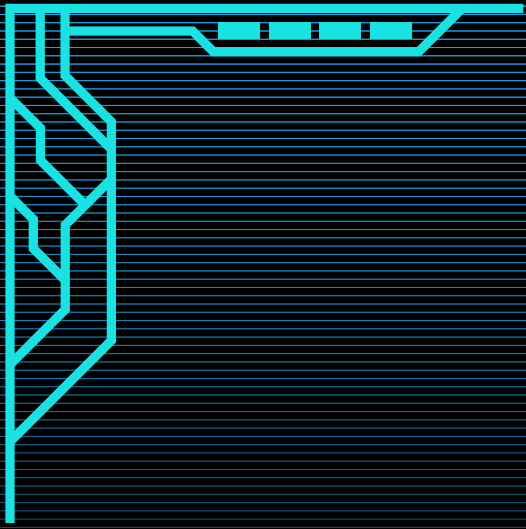


DSA ASSIGNMENT-3



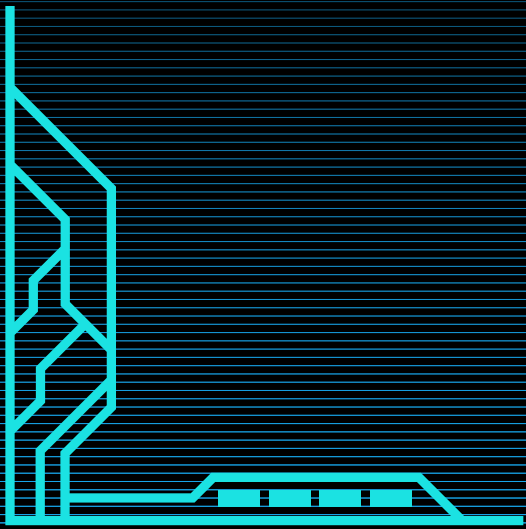
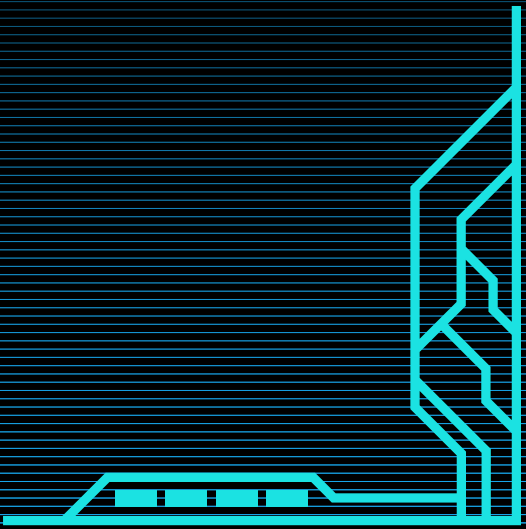
BY: ADHITHIYAN.B
RA2311026050030
GITHUB: [HTTPS://GITHUB
COM/ADHI0007](https://github.com/ADHI0007)





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 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 whether or not the contact is found, along with their position if present.**
- 
- 

main.c



Share

Run

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  struct Node {
6      char name[100];
7      struct Node* next;
8  };
9
10 // Function prototypes
11 void createList(struct Node** head);
12 void insertContact(struct Node** head, char* name, int position);
13 void deleteContact(struct Node** head, char* name, int position, char byName);
14 void displayList(struct Node* head);
15 int searchContact(struct Node* head, char* name);
16
17 int main() {
18     struct Node* head = NULL;
19     int choice, position;
20     char name[100];
21     char byName;
22
23     while (1) {
24         printf("\n1. Create the list of contacts\n");
25         printf("2. Insert a new contact\n");
26         printf("3. Delete a contact\n");
27         printf("4. Display contact list\n");
28         printf("5. Search for a contact\n");
29         printf("6. Exit\n");
30         printf("Enter your choice: ");
31         scanf("%d", &choice);
32
33         switch (choice) {
34             case 1:
35                 createList(&head);
36                 break;
37             case 2:
38                 printf("Enter the contact's name to insert: ");
39                 scanf("%s", name);
40                 printf("Enter the position (0-based index) to insert the contact: ");
41                 scanf("%d", &position);
42                 insertContact(&head, name, position);
43                 displayList(head);
44                 break;
45             case 3:
46                 printf("Delete by name or position? (n/p): ");
47                 scanf(" %c", &byName);
48                 if (byName == 'n') {
49                     printf("Enter the contact's name to delete: ");
50                     scanf("%s", name);
51                     deleteContact(&head, name, -1, 1);
52                 } else {
53                     printf("Enter the position (0-based index) to delete the contact: ");
54                     scanf("%d", &position);
```

```

55         deleteContact(&head, NULL, position, 0);
56     }
57     displayList(head);
58     break;
59     case 4:
60         displayList(head);
61         break;
62     case 5:
63         printf("Enter the contact's name to search: ");
64         scanf("%s", name);
65         position = searchContact(head, name);
66         if (position != -1) {
67             printf("%s found at position %d\n", name, position);
68         } else {
69             printf("%s not found\n", name);
70         }
71         break;
72     case 6:
73         printf("Exiting the program...\n");
74         exit(0);
75         break;
76     default:
77         printf("Invalid choice, try again.\n");
78 }
79
80 return 0;
81 }
82
83 void createList(struct Node** head) {
84     int n;
85     char name[100];
86     printf("Enter the number of contacts: ");
87     scanf("%d", &n);
88
89     for (int i = 0; i < n; i++) {
90         printf("Enter contact name %d: ", i + 1);
91         scanf("%s", name);
92         insertContact(head, name, i);
93     }
94
95     displayList(*head);
96 }
97
98 void insertContact(struct Node** head, char* name, int position) {
99     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
100     struct Node* current = *head;
101     strcpy(newNode->name, name);
102     newNode->next = NULL;
103
104     if (position == 0) {
105         newNode->next = *head;
106         *head = newNode;
107     } else {
108         for (int i = 0; current != NULL && i < position - 1; i++) {
109             current = current->next;
110         }

```

```
111 -     if (current == NULL) {
112 -         printf("Position out of bounds.\n");
113 -         return;
114 -     }
115 -     newNode->next = current->next;
116 -     current->next = newNode;
117 - }
118 - }
119
120 - void deleteContact(struct Node** head, char* name, int position, char byName) {
121 -     struct Node* temp = *head, *prev = NULL;
122
123 -     if (byName) {
124 -         if (temp != NULL && strcmp(temp->name, name) == 0) {
125 -             *head = temp->next;
126 -             free(temp);
127 -             return;
128 -         }
129 -         while (temp != NULL && strcmp(temp->name, name) != 0) {
130 -             prev = temp;
131 -             temp = temp->next;
132 -         }
133 -         if (temp == NULL) {
134 -             printf("Contact not found.\n");
135 -             return;
136 -         }
137 -         prev->next = temp->next;
138 -         free(temp);
139 -     } else {
140 -         if (position == 0 && temp != NULL) {
141 -             *head = temp->next;
142 -             free(temp);
143 -             return;
144 -         }
145 -         for (int i = 0; temp != NULL && i < position; i++) {
146 -             prev = temp;
147 -             temp = temp->next;
148 -         }
149 -         if (temp == NULL) {
150 -             printf("Position out of bounds.\n");
151 -             return;
152 -         }
153 -         prev->next = temp->next;
154 -         free(temp);
155 -     }
156 - }
157
158 - void displayList(struct Node* head) {
```

```
158 void displayList(struct Node* head) {
159     struct Node* temp = head;
160     printf("Contact list: ");
161     while (temp != NULL) {
162         printf("%s -> ", temp->name);
163         temp = temp->next;
164     }
165     printf("NULL\n");
166 }
167
168 int searchContact(struct Node* head, char* name) {
169     struct Node* temp = head;
170     int position = 0;
171
172     while (temp != NULL) {
173         if (strcmp(temp->name, name) == 0) {
174             return position;
175         }
176         temp = temp->next;
177         position++;
178     }
```

Output

Clear

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

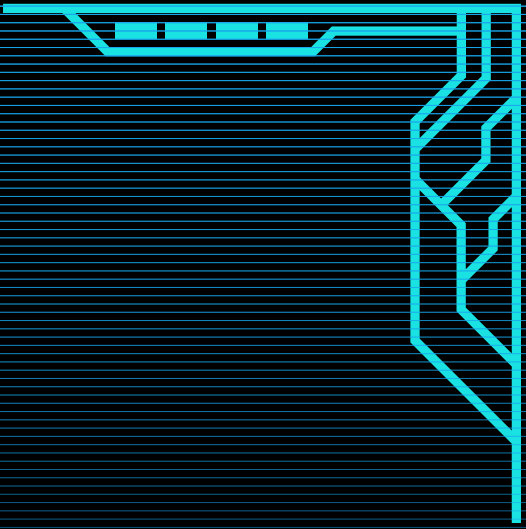
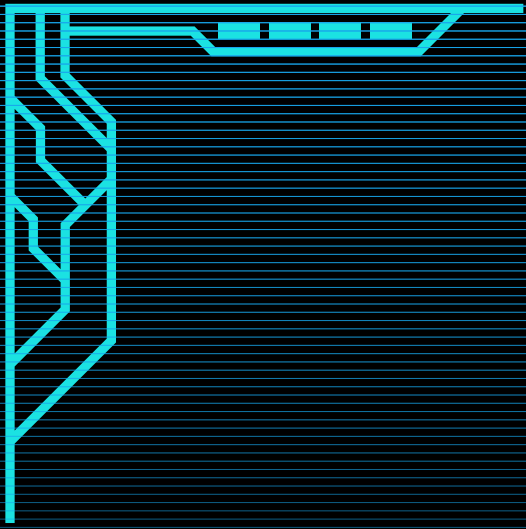
Enter the contact's name to search: Guru

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

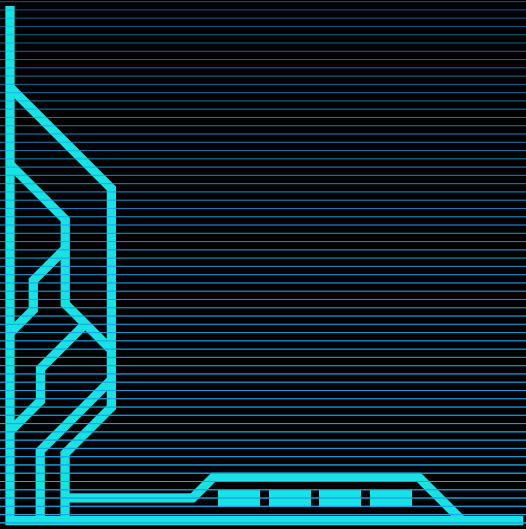
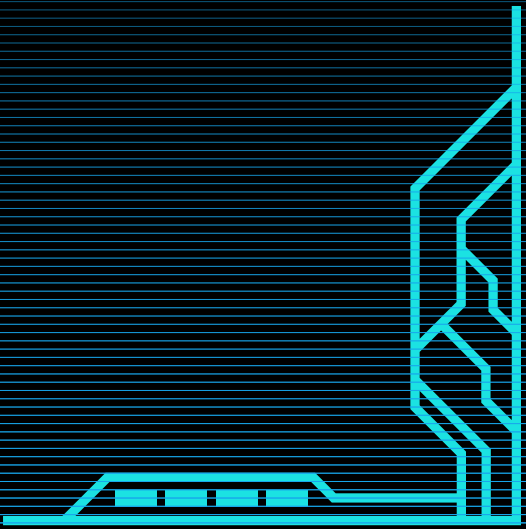
Enter your choice: 6

Exiting the program...

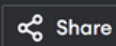


2. You are tasked with implementing a simple contact management system using a doubly linked list. The system will manage contact names. Implement the following operations using a doubly linked list and switch-case. After every operation, display the current list of contacts.

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.**
- 
- 

main.c



Run

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  struct Node {
6      char name[100];
7      struct Node* next;
8      struct Node* prev;
9  };
10
11 // Function prototypes
12 void createList(struct Node** head);
13 void insertContact(struct Node** head, char* name, int position);
14 void deleteContact(struct Node** head, char* name, int position, char byName);
15 void displayListForward(struct Node* head);
16 void displayListBackward(struct Node* head);
17 int searchContact(struct Node* head, char* name);
18
19 int main() {
20     struct Node* head = NULL;
21     int choice, position;
22     char name[100];
23     char byName;
24
25     while (1) {
26         printf("\n1. Create the list of contacts\n");
27         printf("2. Insert a new contact\n");
28         printf("3. Delete a contact\n");
29         printf("4. Display contact list\n");
30         printf("5. Search for a contact\n");
31         printf("6. Exit\n");
32         printf("Enter your choice: ");
33         scanf("%d", &choice);
34
35         switch (choice) {
36             case 1:
37                 createList(&head);
38                 break;
39             case 2:
40                 printf("Enter the contact's name to insert: ");
41                 scanf("%s", name);
42                 printf("Enter the position (0-based index) to insert the contact: ");
43                 scanf("%d", &position);
44                 insertContact(&head, name, position);
45                 displayListForward(head);
46                 displayListBackward(head);
47                 break;
48             case 3:
49                 printf("Delete by name or position? (n/p): ");
50                 scanf(" %c", &byName);
51                 if (byName == 'n') {
52                     printf("Enter the contact's name to delete: ");
53                     scanf("%s", name);
54                     deleteContact(&head, name, -1, 1);
55                 } else {
56                     printf("Enter the position (0-based index) to delete the contact: ");
57                     scanf("%d", &position);
58                     deleteContact(&head, NULL, position, 0);
59                 }
60             }
61         }
62     }
```

```

60         displayListForward(head);
61         displayListBackward(head);
62         break;
63     case 4:
64         displayListForward(head);
65         displayListBackward(head);
66         break;
67     case 5:
68         printf("Enter the contact's name to search: ");
69         scanf("%s", name);
70         position = searchContact(head, name);
71         if (position != -1) {
72             printf("%s found at position %d\n", name, position);
73         } else {
74             printf("%s not found\n", name);
75         }
76         break;
77     case 6:
78         printf("Exiting the program...\n");
79
80         break;
81     default:
82         printf("Invalid choice, try again.\n");
83 }
84 }
85 return 0;
86 }
87
88 void createList(struct Node** head) {
89     int n;
90     char name[100];
91     printf("Enter the number of contacts: ");
92     scanf("%d", &n);
93
94     for (int i = 0; i < n; i++) {
95         printf("Enter contact name %d: ", i + 1);
96         scanf("%s", name);
97         insertContact(head, name, i);
98     }
99
100     displayListForward(*head);
101     displayListBackward(*head);
102 }
103
104 void insertContact(struct Node** head, char* name, int position) {
105     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
106     struct Node* current = *head;
107     strcpy(newNode->name, name);
108     newNode->next = NULL;
109     newNode->prev = NULL;
110
111     if (position == 0) {
112         newNode->next = *head;
113         if (*head != NULL) {
114             (*head)->prev = newNode;
115         }
116         *head = newNode;
117     } else {
118         for (int i = 0; current != NULL && i < position - 1; i++) {

```

```

119         current = current->next;
120     }
121     if (current == NULL) {
122         printf("Position out of bounds.\n");
123         return;
124     }
125     newNode->next = current->next;
126     if (current->next != NULL) {
127         current->next->prev = newNode;
128     }
129     current->next = newNode;
130     newNode->prev = current;
131 }
132 }
133
134 void deleteContact(struct Node** head, char* name, int position, char byName) {
135     struct Node* temp = *head;
136
137     if (byName) {
138         while (temp != NULL && strcmp(temp->name, name) != 0) {
139             temp = temp->next;
140         }
141         if (temp == NULL) {
142             printf("Contact not found.\n");
143             return;
144         }
145     } else {
146         for (int i = 0; temp != NULL && i < position; i++) {
147             temp = temp->next;
148         }
149         if (temp == NULL) {
150             printf("Position out of bounds.\n");
151             return;
152         }
153     }
154
155     if (*head == temp) {
156         *head = temp->next;
157     }
158     if (temp->next != NULL) {
159         temp->next->prev = temp->prev;
160     }
161     if (temp->prev != NULL) {
162         temp->prev->next = temp->next;
163     }
164
165     free(temp);
166 }
167
168 void displayListForward(struct Node* head) {
169     struct Node* temp = head;
170     printf("Contact list (forward): ");
171     while (temp != NULL) {
172         printf("%s <-> ", temp->name);
173         temp = temp->next;
174     }
175     printf("NULL\n");
176 }
177
178 void displayListBackward(struct Node* head) {
179     struct Node* temp = head;
180     if (temp == NULL) {
181         printf("Contact list (backward): NULL\n");

```

```
181     printf("Contact list (backward): NULL\n");
182     return;
183 }
184
185 while (temp->next != NULL) {
186     temp = temp->next;
187 }
188
189 printf("Contact list (backward): ");
190 while (temp != NULL) {
191     printf("%s <-> ", temp->name);
192     temp = temp->prev;
193 }
194 printf("NULL\n");
195 }
196
197 int searchContact(struct Node* head, char* name) {
198     struct Node* temp = head;
199     int position = 0;
200
201     while (temp != NULL) {
202         if (strcmp(temp->name, name) == 0) {
203             return position;
204         }
205         temp = temp->next;
206         position++;
207     }
```

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...
```