



---

## DSA ASSIGNMENT 3

---

PARVATHY OK

RA2311026050020

**QUESTION 1: -**

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 // Structure to represent a node in the linked list
6 typedef struct Node {
7     char name[50];
8     struct Node* next;
9 } Node;
10
11 // Function to create a new node
12 Node* createNode(char* name) {
13     Node* newNode = (Node*) malloc(sizeof(Node));
14     strcpy(newNode->name, name);
15     newNode->next = NULL;
16     return newNode;
17 }
18
19 // Function to create the list of contacts
20 void createList(Node** head) {
21     int numContacts;
22     printf("Enter the number of contacts: ");
23     scanf("%d", &numContacts);
24     for (int i = 0; i < numContacts; i++) {
25         char name[50];
26         printf("Enter contact %d name: ", i + 1);
27         scanf("%s", name);
28         Node* newNode = createNode(name);
29         if (*head == NULL) {
30             *head = newNode;
31         } else {
32             Node* temp = *head;
33             while (temp->next != NULL) {
34                 temp = temp->next;
35             }
36             temp->next = newNode;
37         }
38     }
39 }
```

```
25     char name[50];
26     printf("Enter contact %d name: ", i + 1);
27     scanf("%s", name);
28     Node* newNode = createNode(name);
29     if (*head == NULL) {
30         *head = newNode;
31     } else {
32         Node* temp = *head;
33         while (temp->next != NULL) {
34             temp = temp->next;
35         }
36         temp->next = newNode;
37     }
38 }
39 printf("List created successfully!\n");
40 }
41
42 // Function to insert a new contact
43 void insertContact(Node** head, int position, char* name) {
44     Node* newNode = createNode(name);
45     if (position == 0) {
46         newNode->next = *head;
47         *head = newNode;
48     } else {
49         Node* temp = *head;
50         for (int i = 0; i < position - 1; i++) {
51             temp = temp->next;
```

```
50         for (int i = 0, i < position - 1, i++) {
51             temp = temp->next;
52             if (temp == NULL) {
53                 printf("Invalid position. Please try again.\n");
54                 return;
55             }
56         }
57         newNode->next = temp->next;
58         temp->next = newNode;
59     }
60     printf("Contact inserted successfully!\n");
61 }
62
63 // Function to delete a contact by name
64 void deleteContactByName(Node** head, char* name) {
65     Node* temp = *head;
66     Node* prev = NULL;
67     while (temp != NULL) {
68         if (strcmp(temp->name, name) == 0) {
69             if (prev == NULL) {
70                 *head = temp->next;
71             } else {
72                 prev->next = temp->next;
73             }
74             free(temp);
75             printf("Contact deleted successfully!\n");
76 }
```

```
75         printf("Contact deleted successfully!\n");
76     }
77 }
78 prev = temp;
79 temp = temp->next;
80 }
81 printf("Contact not found.\n");
82 }
83
84 // Function to delete a contact by position
85 void deleteContactByPosition(Node** head, int position) {
86     if (position == 0 && *head != NULL) {
87         Node* temp = *head;
88         *head = (*head)->next;
89         free(temp);
90         printf("Contact deleted successfully!\n");
91         return;
92     }
93     Node* temp = *head;
94     Node* prev = NULL;
95     for (int i = 0; i < position; i++) {
96         prev = temp;
97         temp = temp->next;
98     }
99     if (temp == NULL) {
100         printf("Invalid position. Please try again.\n");
101         return;
```

```
99         printf("Invalid position. Please try again.\n");
100    }
101    }
102    }
103    prev->next = temp->next;
104    free(temp);
105    printf("Contact deleted successfully!\n");
106 }
107
108 // Function to traverse the list
109 void traverseList(Node* head) {
110     printf("Current list of contacts:\n");
111     int position = 1;
112     while (head != NULL) {
113         printf("%d. %s\n", position, head->name);
114         head = head->next;
115         position++;
116     }
117 }
118
119 // Function to search for a contact
120 void searchContact(Node* head, char* name) {
121     int position = 1;
122     while (head != NULL) {
123         if (strcmp(head->name, name) == 0) {
124             printf("Contact found at position %d.\n", position);
125         }
126     }
127 }
```

```
124         printf("Contact found at position %d.\n", position);
125         return;
126     }
127     head = head->next;
128     position++;
129 }
130 printf("Contact not found.\n");
131 }
132
133 // Function to free the list
134 void freeList(Node* head) {
135     Node* temp;
136     while (head != NULL) {
137         temp = head;
138         head = head->next;
139         free(temp);
140     }
141     printf("List memory freed successfully.\n");
142 }
143
144 int main() {
145     Node* head = NULL;
146     while (1) {
147         printf("\nContact Management System Menu:\n");
148         printf("1. Create list\n");
149         printf("2. Insert contact\n");
```

```
149     printf("1. Create contact\n");
150     printf("2. Insert contact\n");
151     printf("3. Delete contact\n");
152     printf("4. Traverse list\n");
153     printf("5. Search contact\n");
154     printf("6. Exit\n");
155     int choice;
156     printf("Enter your choice: ");
157     scanf("%d", &choice);
158     switch (choice) {
159         case 1:
160             createList(&head);
161             traverseList(head);
162             break;
163         case 2: {
164             char name[50];
165             int position;
166             printf("Enter the new contact's name: ");
167             scanf("%s", name);
168             printf("Enter the position to insert the contact: ");
169             scanf("%d", &position);
170             insertContact(&head, position - 1, name);
171             traverseList(head);
172             break;
173         }
174     }
```

```
172         }
173     case 3: {
174         char choice;
175         printf("Do you want to delete by name or by
176             position? (name/position): ");
177         scanf(" %c", &choice);
178         if (choice == 'n' || choice == 'N') {
179             char name[50];
180             printf("Enter the contact's name to delete: ");
181             scanf("%s", name);
182             deleteContactByName(&head, name);
183         } else if (choice == 'p' || choice == 'P') {
184             int position;
185             printf("Enter the position to delete the
186                 contact: ");
187             scanf("%d", &position);
188             deleteContactByPosition(&head, position - 1);
189         } else {
190             printf("Invalid choice. Please try again.\n");
191         }
192         traverseList(head);
193         break;
194     case 4:
195         traverseList(head);
196         break;
197     case 5: {
198         char name[50];
199         printf("Enter the contact's name to search: ");
200         scanf("%s", name);
201         searchContact(head, name);
202         break;
203     case 6:
204         freeList(head);
205         return 0;
206     default:
207         printf("Invalid choice. Please try again.\n");
208     }
209 }
210 return 0;
211 }
212 }
```

## SAMPLE OUTPUT:-

```
/tmp/0CXC5JExu5.o

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 1
Enter the number of contacts: 3
Enter contact 1 name: john
Enter contact 2 name: alice
Enter contact 3 name: bob
List created successfully!
Current list of contacts:
1. john
2. alice
3. bob

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
```

```
5. Search contact
```

```
6. Exit
```

```
Enter your choice: 2
```

```
Enter the new contact's name: sarah
```

```
Enter the position to insert the contact: 2
```

```
Contact inserted successfully!
```

```
Current list of contacts:
```

```
1. john
```

```
2. sarah
```

```
3. alice
```

```
4. bob
```

```
Contact Management System Menu:
```

```
1. Create list
```

```
2. Insert contact
```

```
3. Delete contact
```

```
4. Traverse list
```

```
5. Search contact
```

```
6. Exit
```

```
Enter your choice: 3
```

```
Do you want to delete by name or by position? (name/position): name
```

```
Enter the contact's name to delete: Contact not found.
```

```
Current list of contacts:
```

```
1. john
```

```
2. sarah
```

```
3. alice
```

```
2. sarah  
3. alice  
4. bob
```

```
Contact Management System Menu:
```

- 1. Create list
- 2. Insert contact
- 3. Delete contact
- 4. Traverse list
- 5. Search contact
- 6. Exit

```
Enter your choice: 4
```

```
Current list of contacts:
```

- 1. john
- 2. sarah
- 3. alice
- 4. bob

```
Contact Management System Menu:
```

- 1. Create list
- 2. Insert contact
- 3. Delete contact
- 4. Traverse list
- 5. Search contact
- 6. Exit

```
Enter your choice: 5
```

```
Enter your choice: 5
```

```
Enter the contact's name to search: bob
```

```
Contact found at position 4.
```

```
Contact Management System Menu:
```

- 1. Create list
- 2. Insert contact
- 3. Delete contact
- 4. Traverse list
- 5. Search contact
- 6. Exit

```
Enter your choice: 6
```

```
List memory freed successfully.
```

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

**QUESTION 2:-**

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 // Structure to represent a node in the doubly linked list
6 typedef struct Node {
7     char name[50];
8     struct Node* prev;
9     struct Node* next;
10 } Node;
11
12 // Function to create a new node
13 Node* createNode(char* name) {
14     Node* newNode = (Node*) malloc(sizeof(Node));
15     strcpy(newNode->name, name);
16     newNode->prev = NULL;
17     newNode->next = NULL;
18     return newNode;
19 }
20
21 // Function to create the list of contacts
22 void createList(Node** head, Node** tail) {
23     int numContacts;
24     printf("Enter the number of contacts: ");
25     scanf("%d", &numContacts);
26     for (int i = 0; i < numContacts; i++) {
--
```

```
26     for (int i = 0; i < numContacts; i++) {
27         char name[50];
28         printf("Enter contact %d name: ", i + 1);
29         scanf("%s", name);
30         Node* newNode = createNode(name);
31         if (*head == NULL) {
32             *head = *tail = newNode;
33         } else {
34             (*tail)->next = newNode;
35             newNode->prev = *tail;
36             *tail = newNode;
37         }
38     }
39     printf("List created successfully!\n");
40 }
41
42 // Function to insert a new contact
43 void insertContact(Node** head, Node** tail, int position, char*
    name) {
44     Node* newNode = createNode(name);
45     if (position == 0) {
46         newNode->next = *head;
47         if (*head != NULL) {
48             (*head)->prev = newNode;
49         }
50         *head = newNode;
```

```
49      }
50      *head = newNode;
51      if (*tail == NULL) {
52          *tail = newNode;
53      }
54  } else {
55      Node* temp = *head;
56      for (int i = 0; i < position - 1; i++) {
57          temp = temp->next;
58      if (temp == NULL) {
59          printf("Invalid position. Please try again.\n");
60          free(newNode);
61          return;
62      }
63  }
64  newNode->next = temp->next;
65  newNode->prev = temp;
66  if (temp->next != NULL) {
67      temp->next->prev = newNode;
68  } else {
69      *tail = newNode;
70  }
71  temp->next = newNode;
72 }
73 printf("Contact inserted successfully!\n");
74 }
```

```
74  }
75
76 // Function to delete a contact by name
77 void deleteContactByName(Node** head, Node** tail, char* name) {
78     Node* temp = *head;
79     while (temp != NULL) {
80         if (strcmp(temp->name, name) == 0) {
81             if (temp->prev != NULL) {
82                 temp->prev->next = temp->next;
83             } else {
84                 *head = temp->next;
85             }
86             if (temp->next != NULL) {
87                 temp->next->prev = temp->prev;
88             } else {
89                 *tail = temp->prev;
90             }
91             free(temp);
92             printf("Contact deleted successfully!\n");
93             return;
94         }
95         temp = temp->next;
96     }
97     printf("Contact not found.\n");
98 }
99
```

```
99
100 // Function to delete a contact by position
101 void deleteContactByPosition(Node** head, Node** tail, int position
102     ) {
103     Node* temp = *head;
104     if (position == 0 && *head != NULL) {
105         *head = (*head)->next;
106         if (*head != NULL) {
107             (*head)->prev = NULL;
108         } else {
109             *tail = NULL;
110         }
111         free(temp);
112         printf("Contact deleted successfully!\n");
113     }
114     for (int i = 0; i < position; i++) {
115         temp = temp->next;
116         if (temp == NULL) {
117             printf("Invalid position. Please try again.\n");
118             return;
119         }
120     }
121     if (temp->prev != NULL) {
122         temp->prev->next = temp->next;
123     }
```

```
122     |     temp->prev->next = temp->next;
123     |
124 -    if (temp->next != NULL) {
125     |     temp->next->prev = temp->prev;
126 -    } else {
127     |     *tail = temp->prev;
128     }
129     free(temp);
130     printf("Contact deleted successfully!\n");
131 }
132
133 // Function to traverse the list in forward direction
134 void traverseListForward(Node* head) {
135     printf("Current list of contacts (Forward Traversal):\n");
136     int position = 1;
137 -    while (head != NULL) {
138     |     printf("%d. %s\n", position, head->name);
139     |     head = head->next;
140     |     position++;
141 }
142
143
144 // Function to traverse the list in reverse direction
145 void traverseListReverse(Node* tail) {
146     printf("Current list of contacts (Reverse Traversal):\n");
147     int position = 1;
```

```
147     int position = 1;
148     while (tail != NULL) {
149         printf("%d. %s\n", position, tail->name);
150         tail = tail->prev;
151         position++;
152     }
153 }
154
155 // Function to search for a contact
156 void searchContact(Node* head, char* name) {
157     int position = 1;
158     while (head != NULL) {
159         if (strcmp(head->name, name) == 0) {
160             printf("Contact found at position %d.\n", position);
161             return;
162         }
163         head = head->next;
164         position++;
165     }
166     printf("Contact not found.\n");
167 }
168
169 // Function to free the list
170 void freeList(Node* head) {
171     Node* temp;
172     while (head != NULL) {
```

```
171     Node* temp;
172     while (head != NULL) {
173         temp = head;
174         head = head->next;
175         free(temp);
176     }
177     printf("List memory freed successfully.\n");
178 }
179
180 int main() {
181     Node* head = NULL;
182     Node* tail = NULL;
183     while (1) {
184         printf("\nContact Management System Menu:\n");
185         printf("1. Create list\n");
186         printf("2. Insert contact\n");
187         printf("3. Delete contact\n");
188         printf("4. Traverse list\n");
189         printf("5. Search contact\n");
190         printf("6. Exit\n");
191         int choice;
192         printf("Enter your choice: ");
193         scanf("%d", &choice);
194         switch (choice) {
195             case 1:
196                 createList(&head, &tail);
```

```
195         case 1:
196             createList(&head, &tail);
197             traverseListForward(head);
198             traverseListReverse(tail);
199             break;
200         case 2: {
201             char name[50];
202             int position;
203             printf("Enter the new contact's name: ");
204             scanf("%s", name);
205             printf("Enter the position to insert the contact: "
206                   );
207             scanf("%d", &position);
208             insertContact(&head, &tail, position - 1, name);
209             traverseListForward(head);
210             traverseListReverse(tail);
211             break;
212         }
213         case 3: {
214             char choice;
215             printf("Do you want to delete by name or by
216                   position? (name/position): ");
217             scanf(" %c", &choice);
218             if (choice == 'n' || choice == 'N') {
219                 char name[50];
220                 printf("Enter the contact's name to delete: ");
221                 scanf("%s" name);
```

```
218         printf("Enter the contact's name to delete: ");
219         scanf("%s", name);
220         deleteContactByName(&head, &tail, name);
221     } else if (choice == 'p' || choice == 'P') {
222         int position;
223         printf("Enter the position to delete the
224             contact: ");
225         scanf("%d", &position);
226         deleteContactByPosition(&head, &tail, position
227             - 1);
228     } else {
229         printf("Invalid choice. Please try again.\n");
230     }
231     traverseListForward(head);
232     traverseListReverse(tail);
233     break;
234 }
235 case 4:
236     traverseListForward(head);
237     traverseListReverse(tail);
238     break;
239 case 5: {
240     char name[50];
241     printf("Enter the contact's name to search: ");
242     scanf("%s", name);
243     searchContact(head, name);
244     searchContact(head, name);
245     break;
246 }
247 case 6:
248     freeList(head);
249     return 0;
250 default:
251     printf("Invalid choice. Please try again.\n");
252 }
253
254 }
```

## SAMPLE OUTPUT:-

```
/tmp/mAJHus5JsX.o

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 1
Enter the number of contacts: 3
Enter contact 1 name: john
Enter contact 2 name: alice'
Enter contact 3 name: bob
List created successfully!
Current list of contacts (Forward Traversal):
1. john
2. alice
3. bob
Current list of contacts (Reverse Traversal):
1. bob
2. alice
3. john

Contact Management System Menu:
1. Create list
```

```
Contact Management System Menu:  
1. Create list  
2. Insert contact  
3. Delete contact  
4. Traverse list  
5. Search contact  
6. Exit  
Enter your choice: 2  
Enter the new contact's name: sarah  
Enter the position to insert the contact: 2  
Contact inserted successfully!  
Current list of contacts (Forward Traversal):  
1. john  
2. sarah  
3. alice  
4. bob  
Current list of contacts (Reverse Traversal):  
1. bob  
2. alice  
3. sarah  
4. john  
  
Contact Management System Menu:  
1. Create list  
2. Insert contact  
3. Delete contact  
4. Traverse list
```

```
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 3
Do you want to delete by name or by position? (name/position): p
Enter the position to delete the contact: 1
Contact deleted successfully!
Current list of contacts (Forward Traversal):
1. sarah
2. alice
3. bob
Current list of contacts (Reverse Traversal):
1. bob
2. alice
3. sarah

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 4
Current list of contacts (Forward Traversal):
```

```
| Current list of contacts (Forward Traversal):
```

- 1. sarah
- 2. alice
- 3. bob

```
| Current list of contacts (Reverse Traversal):
```

- 1. bob
- 2. alice
- 3. sarah

```
| Contact Management System Menu:
```

- 1. Create list
- 2. Insert contact
- 3. Delete contact
- 4. Traverse list
- 5. Search contact
- 6. Exit

```
| Enter your choice: 5
```

```
| Enter the contact's name to search: bob
```

```
| Contact found at position 3.
```

```
| Contact Management System Menu:
```

- 1. Create list
- 2. Insert contact
- 3. Delete contact
- 4. Traverse list
- 5. Search contact
- 5. Search contact
- 6. Exit

```
| Enter your choice: 6
```

```
| List memory freed successfully.
```

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