

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct node {
5      int data;
6      struct node *next;
7  };
8  struct node *head = NULL;
9
10 void createList(int n) {
11     struct node *newNode, *temp = NULL;
12     int data;
13     printf("Enter %d elements:\n", n);
14     for (int i = 1; i <= n; i++) {
15         newNode = malloc(sizeof(struct node));
16         if (!newNode) {
17             printf("Memory allocation failed.\n");
18             return;
19         }
20         printf("Element %d: ", i);
21         scanf("%d", &data);
22         newNode->data = data;
23         newNode->next = NULL;
24         if (head == NULL)
25             head = temp = newNode;
26         else {
27             temp->next = newNode;
28             temp = newNode;
29         }
30     }
31     printf("List created successfully.\n");
32 }
33
34 void deleteAtBeginning() {
35     if (head == NULL) {
36         printf("List is empty\n"); return;
37     }
38     struct node* temp = head;
39     head = head->next;
40     free(temp);
41 }
42
43 void deleteAtEnd() {
44     if (head == NULL) {printf("List is empty\n"); return;}
45     if (head->next == NULL) {
46         free(head);
47         head = NULL; return;
48     }
49     struct node *temp;
50     struct node *tail;
51     temp = head;
52     while (temp->next != NULL) {
53         tail = temp;
54         temp = temp->next;
55     }
56     free(temp);
57     tail->next = NULL;
58 }

```

```

54     }
55     tail->next = NULL;
56     free(temp);
57 }
58 void deleteAtanyPos(int pos){
59     if(head == NULL){printf("List is empty\n");return;}
60     if(pos == 1){deleteAtbeginning();return;}
61     struct node *temp,*tail;
62     temp=head;int i=1;
63     while(i<pos-1 && temp!=NULL){
64         tail = temp;
65         temp = temp->next;i++;
66     }
67     if(temp == NULL | temp->next == NULL){
68         printf("Invalid position\n");return;
69     }
70     tail->next=temp->next;
71     free(temp);
72 }
73 void displayList() {
74     struct node *temp = head;
75     if (!head) {
76         printf("List is empty.\n");
77         return;
78     }
79     printf("Current List: ");
80     while (temp) {
81         printf("%d -> ", temp->data);
82         temp = temp->next;
83     }
84     printf("NULL\n");
85 }
86
87 int main() {
88     int choice, n, data, pos;
89
90     while (1) {
91         printf("\n---- Singly Linked List operation ----\n");
92         printf("1. Create List\n");
93         printf("2. Delete at Beginning\n");
94         printf("3. Delete at Position\n");
95         printf("4. Delete at End\n");
96         printf("5. Display List\n");
97         printf("6. Exit\n");
98         printf("Enter your choice: ");
99         scanf("%d", &choice);
100
101     }
102
103 int main() {
104     int choice, n, data, pos;
105
106     while (1) {
107         printf("\n---- Singly Linked List operation ----\n");
108         printf("1. Create List\n");
109         printf("2. Delete at Beginning\n");
110         printf("3. Delete at Position\n");
111         printf("4. Delete at End\n");
112         printf("5. Display List\n");
113         printf("6. Exit\n");
114         printf("Enter your choice: ");
115         scanf("%d", &choice);
116         switch (choice)
117         {
118             case 1:
119                 printf("Enter number of nodes: ");
120                 scanf("%d", &n);
121                 createList(n);break;
122             case 2:
123                 deleteAtbeginning();break;
124             case 3:
125                 printf("Enter position of node: ");
126                 scanf("%d",&pos);
127                 deleteAtanyPos(pos);break;
128             case 4:
129                 deleteAtEnd();break;
130             case 5:
131                 displayList();break;
132             case 6:
133                 printf("Exiting...\n");
134                 return 0;
135             default:
136                 printf("Invalid choice! Try again.\n");break;
137         }
138     }
139 }

```

---- Singly Linked List operation ----

1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 1

Enter number of nodes: 4

Enter 4 elements:

Element 1: 10

Element 2: 20

Element 3: 30

Element 4: 40

List created successfully.

---- Singly Linked List operation ----

1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 2

---- Singly Linked List operation ----

1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 5

Current List: 20 -> 30 -> 40 -> NULL

---- Singly Linked List operation ----

1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 3

Enter position of node: 3

---- Singly Linked List operation ----

1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 5

Current List: 20 -> 40 -> NULL

Lab 5

@ No implement deletion of Node in a linked list as creation of LL b) display the contents of the linked list.
pseudocode:-

```
Function Delete_At_Begin()
    if head == NULL
        print "List is Empty"
        return
    temp = head;
    head = head.next;
    delete temp;
END Function
```

```
Function DELETE_AT_END()
    if head == NULL
        print "List is Empty"
        return
    if head.next == NULL
        delete_At_Begin()
        return
    temp = head
    while temp.next != NULL
        temp = temp.next
    temp.next = NULL
    delete temp
End function.
```

```
Function. delete_at_pos(pos)
    if head == NULL
        print "List is Empty"
        return
    if pos == 1
        delete_At_Begin()
        return
    temp = head
    i = 1
    while i < pos-1 and temp != NULL
        temp = temp.next
        i++
    if temp == NULL or temp.next == NULL
        print "invalid position"
        return
```

```

nodeToDelete = temp->next;
temp->next = nodeToDelete->next;
delete nodeToDelete;

Code:-
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};

struct node * head = NULL;

void createList(int n) {
    struct node * newnode, * temp = NULL;
    int data;
    printf("Enter %d elements: \n", n);
    for (int i = 1; i <= n; i++) {
        newnode = malloc(sizeof(struct node));
        if (!newnode) {
            printf("Memory allocation failed\n"); return;
        }
        printf("Element %d: ", i);
        scanf("%d", &data);
        newnode->data = data;
        newnode->next = NULL;
        if (head == NULL) {
            head = temp = newnode;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
}

```

```

void deleteAtEnd() {
    if (head == NULL) { printf("List is empty\n"); return; }
    if (head->next == NULL) {
        head = NULL;
        return;
    }
    struct node * temp;
    struct node * tail;
    temp = head;
    while (temp->next != NULL) {
        tail = temp;
        temp = temp->next;
    }
    tail->next = NULL;
    free(temp);
}

void deleteAtBeginning() {
    if (head == NULL) { printf("List is empty\n"); return; }
    struct node * temp = head;
    head = head->next;
    free(temp);
}

void displayList() {
    if (head == NULL) { printf("List is empty\n"); return; }
    if (pos == 1) { deleteAtBeginning(); return; }
    struct node * temp, * tail;
    temp = head; int i = 1;
    while (i < pos-1 && temp->next != NULL) {
        temp = temp->next; i++;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position\n"); return;
    }
    tail->next = temp->next;
    free(temp);
}

```

```

void displayList() {
    struct node * temp = head;
    if (head == NULL) { printf("List is empty\n"); return; }
    printf("Current List: ");
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("1. Create List\n");
        printf("2. Delete at Beginning\n");
        printf("3. Delete at position\n");
        printf("4. Delete at End\n");
        printf("5. Display List\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n); break;
            case 2:
                deleteAtBeginning(); break;
            case 3:
                printf("Enter position of node: ");
                scanf("%d", &pos);
                deleteAtPosition(pos); break;
            case 4:
                deleteAtEnd(); break;
            case 5:
                displayList(); break;
            case 6:
                printf("Exiting... \n");
                return 0;
        }
    }
}

```

```

default:
    printf("Invalid choice! Try again\n");
}

O/P:-
---- Singly Linked List Operations ----
1. Create List
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 1
Enter number of nodes: 4
Enter 4 Elements:
Element 1: 20
Element 2: 20
Element 3: 30
Element 4: 40
List created Successfully.

---- Singly Linked List Operations ----
1. Create List
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 2
Current List: 20 -> 30 -> 40 -> NULL

```

----- Singly Linked List operation -----

1. Create List
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 3

Enter position of node: 3

----- Singly Linked List operation -----

1. Create List
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit

Enter your choice: 3

Current List: 20 → 30 → NULL

~~PS:~~