

```
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     struct node * temp = head;
38     head = head->next;
39     free(temp);
40 }
41
42 void deleteAtEnd() {
43     if (head == NULL) {printf("List is empty\n"); return; }
44     if (head->next == NULL) {
45         free(head);
46         head = NULL; return;
47     }
48     struct node *temp;
49     struct node *tail;
50     temp = head;
51     while (temp->next != NULL) {
52         tail = temp;
53         temp = temp->next;
54     }
55     tail->next = NULL;
56 }
```

```

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");
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    int main(){
103        int choice, n, data, pos;
104
105        while (1){
106            printf("\n---- Singly Linked List operation ----\n");
107            printf("1. Create List\n");
108            printf("2. Delete at Beginning\n");
109            printf("3. Delete at Position\n");
110            printf("4. Delete at End\n");
111            printf("5. Display List\n");
112            printf("6. Exit\n");
113            printf("Enter your choice: ");
114            scanf("%d", &choice);
115            switch (choice)
116            {
117                case 1:
118                    printf("Enter number of nodes: ");
119                    scanf("%d", &n);
120                    createList(n);break;
121                case 2:
122                    deleteAtbeginning();break;
123                case 3:
124                    printf("Enter position of node: ");
125                    scanf("%d", &pos);
126                    deleteAtAnyPos(pos);break;
127                case 4:
128                    deleteAtEnd();break;
129                case 5:
130                    displayList();break;
131                case 6:
132                    printf("Exiting...\n");
133                    return 0;
134                default:
135                    printf("Invalid choice! Try again.\n");break;
136            }
137        }
138    }
139 }
140 }
```

```
---- 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

- ① To 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

End while

~~temp = temp.next~~

tail.next = ~~temp~~ NULL

delete temp

End function.

Function. delete\_at\_pos(pos)

if Head == NULL

print "List is Empty"

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-
if (node < 1000)
if (node < 500)
{
    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 element : ", 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 = newNode;
        else
        {
            temp->next = newNode;
            temp = newNode;
        }
    }
}

```

```

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

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

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

```

```

void displayList()
{
    struct node *temp = head;
    if (head == NULL)
        printf("List is empty\n");
    else
    {
        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 nodes : ");
                scanf("%d", &pos);
                deleteAtPos(pos);
                break;
            case 4:
                deleteAtEnd();
                break;
            case 5:
                displayList();
                break;
            case 6:
                printf("Exiting...\n");
                return 0;
        }
    }
}

```

```

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

    0/1' --- Singly Linked List Operations ---
    1. CreateList
    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. CreateList
    2. Delete at Beginning
    3. Delete at position
    4. Delete at End
    5. Display List
    6. Exit

    Enter your choice : 2

```

```

    --- Singly Linked List Operations ---
    1. CreateList
    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, Insert & Delete

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

~~RI~~

a) w & p  
(insert & delete operation)

b) prw  
? 1

----- Insertion at Beginning -----

prw → 20 → 30 → NULL  
prw → 30 → 20 → NULL  
prw → 20 → NULL  
prw → NULL

prw → 30 → NULL  
prw → NULL

----- Insertion at End -----

prw → 20 → 30 → NULL  
prw → 20 → 30 → 40 → NULL

prw → 20 → 30 → 40 → 50 → NULL

prw → 20 → 30 → 40 → 50 → 60 → NULL