

Lab 5

Deletion at Start

Input:

```
#include <stdio.h>
#include <stdlib.h>

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

struct Node* deleteAtStart(struct Node* head) {
    if (head == NULL) return NULL;

    struct Node* temp = head;
    head = head->next;
    free(temp);
    return head;
}

int main() {
    struct Node *n1 = malloc(sizeof(struct Node));
    struct Node *n2 = malloc(sizeof(struct Node));
    struct Node *n3 = malloc(sizeof(struct Node));
    struct Node *n4 = malloc(sizeof(struct Node));

    n1->data = 10; n1->next = n2;
    n2->data = 20; n2->next = n3;
    n3->data = 30; n3->next = n4;
    n4->data = 40; n4->next = NULL;
```

```
n1->data = 10; n1->next = n2;
n2->data = 20; n2->next = n3;
n3->data = 30; n3->next = n4;
n4->data = 40; n4->next = NULL;

struct Node* head = deleteAtStart(n1);
struct Node* curr = head;

printf("%d -> ", curr->data);
curr = curr->next;

printf("%d -> ", curr->data);
curr = curr->next;

printf("%d -> ", curr->data);
curr = curr->next;

printf("NULL\n");

return 0;
```

Output:

```
20 -> 30 -> 40 -> NULL
```

Deletion at End

Input:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5      int data;
6      struct Node* next;
7  };
8
9
10 struct Node* head = NULL;
11
12 void deleteAtEnd() {
13     if (head == NULL) {
14         printf("List is empty\n");
15         return;
16     }
17     if (head->next == NULL) {
18         free(head);
19         head = NULL;
20         return;
21     }
22
23     struct Node *temp = head;
24     while (temp->next->next != NULL) {
25         temp = temp->next;
26     }
27     free(temp->next);
```

10

```
33
34     struct Node *n1 = (struct Node*)malloc(sizeof(struct Node));
35     struct Node *n2 = (struct Node*)malloc(sizeof(struct Node));
36     struct Node *n3 = (struct Node*)malloc(sizeof(struct Node));
37     struct Node *n4 = (struct Node*)malloc(sizeof(struct Node));
38
39
40     n1->data = 10; n1->next = n2;
41     n2->data = 20; n2->next = n3;
42     n3->data = 30; n3->next = n4;
43     n4->data = 40; n4->next = NULL;
44
45     head = n1;
46
47
48     deleteAtEnd();
49
50
51     struct Node* curr = head;
52
53
54     if (curr != NULL) {
55         printf("%d -> ", curr->data);
56         curr = curr->next;
57     }
58
59     if (curr != NULL) {
60         printf("%d -> ", curr->data);
```

```
57     }
58
59     if (curr != NULL) {
60         printf("%d -> ", curr->data);
61         curr = curr->next;
62     }
63
64     if (curr != NULL) {
65         printf("%d -> ", curr->data);
66         curr = curr->next;
67     }
68
69     printf("NULL\n");
70
71     return 0;
72 }
```

Output:

```
10 -> 20 -> 30 -> NULL
=== Code Execution Successful ===
```

Deletion at position

Input:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5      int data;
6      struct Node* next;
7  };
8
9  struct Node* head = NULL;
10
11
12 void deleteAtStart() {
13     if (head == NULL) return;
14     struct Node* temp = head;
15     head = head->next;
16     free(temp);
17 }
18
19 void deleteAtPosition(int position) {
20     if (head == NULL || position < 1) {
21         printf("Invalid Operation\n");
22         return;
23     }
24
25     if (position == 1) {
26         deleteAtStart();
27         return;
28     }
```

```
28     }
29
30     struct Node *temp = head;
31
32     for (int i = 1; i < position - 1 && temp != NULL; i++) {
33         temp = temp->next;
34     }
35
36
37     if (temp == NULL || temp->next == NULL) {
38         printf("Position out of range\n");
39         return;
40     }
41
42     struct Node *delNode = temp->next;
43     temp->next = delNode->next;
44     free(delNode);
45     printf("Deleted node at position %d\n", position);
46 }
47
48 int main() {
49
50     struct Node *n1 = (struct Node*)malloc(sizeof(struct Node));
51     struct Node *n2 = (struct Node*)malloc(sizeof(struct Node));
52     struct Node *n3 = (struct Node*)malloc(sizeof(struct Node));
53     struct Node *n4 = (struct Node*)malloc(sizeof(struct Node));
54
```

```
54
55     n1->data = 10; n1->next = n2;
56     n2->data = 20; n2->next = n3;
57     n3->data = 30; n3->next = n4;
58     n4->data = 40; n4->next = NULL;
59
60     head = n1;
61
62
63     deleteAtPosition(3);
64
65
66     struct Node* curr = head;
67
68     if (curr != NULL) {
69         printf("%d -> ", curr->data);
70         curr = curr->next;
71     }
72     if (curr != NULL) {
73         printf("%d -> ", curr->data);
74         curr = curr->next;
75     }
76     if (curr != NULL) {
77         printf("%d -> ", curr->data);
78         curr = curr->next;
79     }
```



```
6     struct Node* curr = head;
7
8     if (curr != NULL) {
9         printf("%d -> ", curr->data);
10        curr = curr->next;
11    }
12    if (curr != NULL) {
13        printf("%d -> ", curr->data);
14        curr = curr->next;
15    }
16    if (curr != NULL) {
17        printf("%d -> ", curr->data);
18        curr = curr->next;
19    }
20
21    printf("NULL\n");
22
23    return 0;
24 }
```

Output:

```
Deleted node at position 3
10 -> 20 -> 40 -> NULL
```