

Lab-5: Deletion of Node

Code:

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

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

struct Node* head = NULL;

// DELETE AT START
void deleteAtStart() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node *temp = head;
    head = head->next;
    printf("Deleted %d from start.\n", temp->data);
    free(temp);
}

// DELETE AT END
void deleteAtEnd() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    if (head->next == NULL) {
        printf("Deleted %d from end.\n", head->data);
        free(head);
        head = NULL;
        return;
    }
    struct Node* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
```

```

    }

    printf("Deleted %d from end.\n", temp->next->data);
    free(temp->next);
    temp->next = NULL;
}

// DELETE AT SPECIFIC POSITION
void deleteAtPosition(int position) {
    if (head == NULL || position < 1) {
        printf("Invalid operation\n");
        return;
    }

    if (position == 1) {
        deleteAtStart();
        return;
    }

    struct Node *temp = head;
    // Move to the (position - 1) node
    for (int i = 1; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }

    if (temp == NULL || temp->next == NULL) {
        printf("Position out of range\n");
        return;
    }

    struct Node *delNode = temp->next;
    temp->next = delNode->next; // Bypass the node
    printf("Deleted %d from position %d.\n", delNode->data, position);
    free(delNode);
}

// Helper to add data
void insert(int val) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = head;
    head = newNode;
}

```

```
void display() {
    struct Node* t = head;
    while (t) {
        printf("%d -> ", t->data);
        t = t->next;
    }
    printf("NULL\n");
}

int main() {
    insert(40); insert(30); insert(20); insert(10);
    printf("Initial List: "); display(); // 10 -> 20 -> 30 -> 40 ->
NULL

    deleteAtPosition(2);
    display();

    deleteAtEnd();
    display();

    deleteAtStart();
    display();

    return 0;
}
```

Output

Clear

```
Initial List: 10 -> 20 -> 30 -> 40 -> NULL
Deleted 20 from position 2.
10 -> 30 -> 40 -> NULL
Deleted 40 from end.
10 -> 30 -> NULL
Deleted 10 from start.
30 -> NULL
```

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