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

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

void linkedListTraversal(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}

// Case 1: Deleting the first element from the linked list
struct Node * deleteFirst(struct Node * head){
    struct Node * ptr = head;
    head = head->next;
    free(ptr);
    return head;
}

// Case 2: Deleting the element at a given index from the l
struct Node * deleteAtIndex(struct Node * head, int index){
    struct Node *p = head;
    struct Node *q = head->next;
    for (int i = 0; i < index-1; i++)
    {
        p = p->next;
        q = q->next;
    }

    p->next = q->next;
    free(q);
    return head;
```

```
40: }
41:
42: // Case 3: Deleting the last element
43: struct Node * deleteAtLast(struct Node * head){
44:     struct Node *p = head;
45:     struct Node *q = head->next;
46:     while(q->next !=NULL)
47:     {
48:         p = p->next;
49:         q = q->next;
50:     }
51:
52:     p->next = NULL;
53:     free(q);
54:     return head;
55: }
56:
57:
58: // Case 4: Deleting the element with a given value from the
59: struct Node * deleteAtIndex2(struct Node * head, int value){
60:     struct Node *p = head;
61:     struct Node *q = head->next;
62:     while(q->data!=value && q->next!= NULL)
63:     {
64:         p = p->next;
65:         q = q->next;
66:     }
67:
68:     if(q->data == value){
69:         p->next = q->next;
70:         free(q);
71:     }
72:     return head;
73: }
74: int main()
75: {
76:     struct Node *head;
77:     struct Node *second;
78:     struct Node *third;
```

```c
    struct Node *fourth;

    // Allocate memory for nodes in the linked list in Heap
    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    // Link first and second nodes
    head->data = 4;
    head->next = second;

    // Link second and third nodes
    second->data = 3;
    second->next = third;

    // Link third and fourth nodes
    third->data = 8;
    third->next = fourth;

    // Terminate the list at the third node
    fourth->data = 1;
    fourth->next = NULL;

    printf("Linked list before deletion\n");
    linkedListTraversal(head);

    // head = deleteFirst(head); // For deleting first element
    // head = deleteAtIndex(head, 2);
    head = deleteAtLast(head);
    printf("Linked list after deletion\n");
    linkedListTraversal(head);

    return 0;
}
```