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


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


struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}


void append(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

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


void deleteElement(struct Node** head, int key) {
    if (*head == NULL) {
        printf("List is empty.\n");
        return;
    }


    if ((*head)->data == key) {
        struct Node* temp = *head;
        *head = (*head)->next;
        free(temp);
        return;
```

```
    if ((*head)->data == key) {
        struct Node* temp = *head;
        *head = (*head)->next;
        free(temp);
        return;
    }


    struct Node* temp = *head;
    while (temp->next != NULL && temp->next->data != key) {
        temp = temp->next;
    }

    if (temp->next != NULL) {
        struct Node* nodeToDelete = temp->next;
        temp->next = temp->next->next;
        free(nodeToDelete);
    } else {
        printf("Element %d not found in the list.\n", key);
    }
}

void deleteLast(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty.\n");
        return;
    }


    if ((*head)->next == NULL) {
        free(*head);
        *head = NULL;
        return;
    }

    struct Node* temp = *head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    free(temp->next);
    temp->next = NULL;
}

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

```c
    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}


int main() {
    struct Node* head = NULL;
    int choice, value, key;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Create (Append to Linked List)\n");
        printf("2. Delete First Element\n");
        printf("3. Delete Specified Element\n");
        printf("4. Delete Last Element\n");
        printf("5. Display Linked List\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                append(&head, value);
                break;
            case 2:
                deleteFirst(&head);
                break;
            case 3:
                printf("Enter element to delete: ");
                scanf("%d", &key);
                deleteElement(&head, key);
                break;
            case 4:
                deleteLast(&head);
                break;
            case 5:
                displayList(head);
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

```
Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 1
Enter value to insert: 45

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 1
Enter value to insert: 40

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 2

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 5
Linked List: 40 -> NULL

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 3
Enter element to delete: 40

Menu:
```

```
Linked List: 40 -> NULL

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 3
Enter element to delete: 40

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 5
List is empty.

Menu:
1. Create (Append to Linked List)
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display Linked List
6. Exit
Enter your choice: 6

Process returned 0 (0x0)   execution time : 60.118 s
Press any key to continue.
```