Singly Linked List Deletion Operations

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

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

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

void createList(struct Node** head, int value) {
   struct Node* newNode = createNode(value);

   if (*head == NULL) {
      *head = newNode;
      return;
   }

   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);
    printf("First element deleted.\n");
}

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

    struct Node *temp = *head, *prev = NULL;

    if (temp != NULL && temp->data == key) {
        *head = temp->next;
        free(temp);
        printf("Element %d deleted.\n", key);
```

```
        return;
    }

    while (temp != NULL && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Element %d not found!\n", key);
        return;
    }

    prev->next = temp->next;
    free(temp);
    printf("Element %d deleted.\n", key);
}

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

    struct Node *temp = *head, *prev = NULL;

    // Only one node
```

```
    if (temp->next == NULL) {
        free(temp);
        *head = NULL;
        printf("Last element deleted.\n");
        return;
    }

    // Traverse to last node
    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }

    prev->next = NULL;
    free(temp);
    printf("Last element deleted.\n");
}

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

    printf("Linked List: ");
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
```

```
        temp = temp->next;
    }
    printf("NULL\n");
}

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

    while (1) {
        printf("\n--- Singly Linked List Menu ---\n");
        printf("1. Create List (Insert at End)\n");
        printf("2. Delete First Element\n");
        printf("3. Delete Specific Element\n");
        printf("4. Delete Last Element\n");
        printf("5. Display 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);
                createList(&head, value);
                break;

            case 2:
```

```
            deleteFirst(&head);
            break;

        case 3:
            printf("Enter value to delete: ");
            scanf("%d", &value);
            deleteSpecific(&head, value);
            break;

        case 4:
            deleteLast(&head);
            break;

        case 5:
            display(head);
            break;

        case 6:
            exit(0);

        default:
            printf("Invalid choice! Try again.\n");
      }
   }

   return 0;
}
```

```
--- Singly Linked List Menu ---
1. Create List (Insert at End)
2. Delete First Element
3. Delete Specific Element
4. Delete Last Element
5. Display List
6. Exit
Enter your choice: 1
Enter value to insert: 10
Enter your choice: 1
Enter value to insert: 20
Enter your choice: 1
```

```
Enter value to insert: 30
Enter your choice: 1
Enter value to insert: 40
Enter your choice: 1
Enter value to insert: 50
Enter your choice: 2
First element deleted.
Enter your choice: 3
Enter value to delete: 30
Element 30 deleted.
Enter your choice: 4
Last element deleted.
```

```
Enter your choice: 5
Linked List: 20 -> 40 -> NULL
Enter your choice: 6
```