Linked list insertion and deletion

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
	int data;
	struct Node* next;
};
void insertAtBeginning(struct Node** head, int value) {
	struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
	newNode->data = value;
	newNode->next = *head;
	*head = newNode;
}
void insertAtEnd(struct Node** head, int value) {
	struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
	struct Node* temp = *head;
	newNode->data = value;
	newNode->next = NULL;

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

```
    while (temp->next != NULL) {
        temp = temp->next;
    }

    temp->next = newNode;
}
void insertAtPosition(struct Node** head, int value, int position) {
    if (position <= 0) {
        printf("Invalid position\n");
        return;
    }

    if (position == 1 || *head == NULL) {
        insertAtBeginning(head, value);
        return;
    }
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    struct Node* temp = *head;
    int count = 1;

    while (count < position - 1 && temp->next != NULL) {
        temp = temp->next;
        count++;
    }
```

```
    if (count < position - 1) {
        printf("Invalid position\n");
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
}
void displayLinkedList(struct Node* head) {
    struct Node* temp = head;

    if (temp == NULL) {
        printf("Linked list is empty.\n");
        return;
    }

    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }

    printf("NULL\n");
}
void deleteAtBeginning(struct Node** head) {
```

```
    if (*head == NULL) {
        printf("Linked list is already empty.\n");
        return;
    }
    struct Node* temp = *head;
    *head = (*head)->next;
    free(temp);
}
void deleteAtEnd(struct Node** head) {
    if (*head == NULL) {
        printf("Linked list is already empty.\n");
        return;
    }
    struct Node* temp = *head;
    struct Node* prev = NULL;

    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }
    if (prev == NULL) {
        *head = NULL;
    } else {
        prev->next = NULL;
    }
```

```
    free(temp);
}
void deleteAtPosition(struct Node** head, int position) {
    if (*head == NULL) {
        printf("Linked list is already empty.\n");
        return;
    }
    struct Node* temp = *head;
    struct Node* prev = NULL;
    if (position == 1) {
        *head = temp->next;
        free(temp);
        return;
    }
    for (int i = 1; temp != NULL && i < position; i++) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Invalid position.\n");
        return;
    }
    prev->next = temp->next;
    free(temp);
}
```

```
int main() {
    struct Node* head = NULL;
    displayLinkedList(head);
    insertAtBeginning(&head, 1);
    insertAtBeginning(&head, 2);
    insertAtBeginning(&head, 3);
    printf("Linked list after insertion at the beginning: ");
    displayLinkedList(head);
    insertAtEnd(&head, 40);
    insertAtEnd(&head, 50);
    printf("Linked list after insertion at the end: ");
    displayLinkedList(head);
    insertAtPosition(&head, 25, 2);
    insertAtPosition(&head, 35, 4);
    printf("Linked list after insertion at specific positions: ");
    displayLinkedList(head);
    deleteAtBeginning(&head);
    printf("Linked list after deletion at the beginning: ");
    displayLinkedList(head);
    deleteAtEnd(&head);
    printf("Linked list after deletion at end: ");
    displayLinkedList(head);
    deleteAtPosition(&head, 3);
    printf("Linked list after deletion at specified position: ");
    displayLinkedList(head);
```

```
    return 0;
}
```

output