DoublyLinkedlist

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


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


struct Node* head = NULL;


void createList(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;
    newNode->prev = NULL;


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

```
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}

void insertLeft(int key, int value) {
    struct Node* temp = head;

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

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

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;

    newNode->next = temp;
    newNode->prev = temp->prev;
```

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

    printf("Inserted %d to the left of %d\n", value, key);
}

void deleteNode(int key) {
    struct Node* temp = head;

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

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

    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
```

```
    } else {
        head = temp->next;  // Update head if deleting first node
    }

    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }

    free(temp);
    printf("Deleted node with value %d\n", key);
}

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

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

```
    printf("\n");
}

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

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

    printf("Doubly Linked List (Reverse): ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

int main() {
    int choice, value, key, cont = 1;
```

```
printf("=== Doubly Linked List Operations ===\n");

while (cont) {
    printf("\n--- MENU ---\n");
    printf("1. Create Node\n");
    printf("2. Insert Left of Node\n");
    printf("3. Delete Node\n");
    printf("4. Display Forward\n");
    printf("5. Display Reverse\n");
    printf("0. Exit\n");
    printf("Enter choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter value to create: ");
            scanf("%d", &value);
            createList(value);
            printf("Created node with %d\n", value);
            break;

        case 2:
            printf("Enter key value (insert left of): ");
```

```
    scanf("%d", &key);
    printf("Enter value to insert: ");
    scanf("%d", &value);
    insertLeft(key, value);
    break;

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

case 4:
    displayList();
    break;

case 5:
    displayReverse();
    break;

case 0:
    cont = 0;
    break;
```

```
            default:
                printf("Invalid choice!\n");
        }
    }

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

    printf("Program terminated.\n");
    return 0;
}
```