WAP to Implement doubly link list with primitive operations

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

// Node structure for doubly linked list
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

// Function to create a new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a new node to the left of a specific node
void insertLeft(struct Node** head, int value, int targetValue) {
    struct Node* newNode = createNode(value);
    struct Node* current = *head;

    while (current != NULL) {
        if (current->data == targetValue) {
            if (current->prev == NULL) {
                // Insert at the beginning
                newNode->next = current;
                current->prev = newNode;
                *head = newNode;
            } else {
                // Insert in the middle
                newNode->next = current;
                newNode->prev = current->prev;
                current->prev->next = newNode;
                current->prev = newNode;
            }
            printf("Node inserted to the left of %d.\n", targetValue);
            return;
        }
        current = current->next;
    }

    printf("Node with value %d not found.\n", targetValue);
```

```
}

// Function to delete a node based on specific value
void deleteNode(struct Node** head, int value) {
    struct Node* current = *head;

    while (current != NULL) {
        if (current->data == value) {
            if (current->prev == NULL) {
                // Delete at the beginning
                *head = current->next;
                if (*head != NULL)
                    (*head)->prev = NULL;
            } else {
                // Delete in the middle
                current->prev->next = current->next;
                if (current->next != NULL)
                    current->next->prev = current->prev;
            }
            free(current);
            printf("Node with value %d deleted.\n", value);
            return;
        }
        current = current->next;
    }

    printf("Node with value %d not found.\n", value);
}

// Function to display the contents of the doubly linked list
void displayList(struct Node* head) {
    struct Node* current = head;

    printf("Doubly Linked List: ");
    while (current != NULL) {
        printf("%d <-> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

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

    do {
        printf("\n1. Create Doubly Linked List\n");
        printf("2. Insert Node to the Left\n");
        printf("3. Delete Node\n");
        printf("4. Display List\n");
```

```
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter the value to create the list: ");
            scanf("%d", &value);
            head = createNode(value);
            printf("Doubly Linked List created.\n");
            break;

        case 2:
            if (head == NULL) {
                printf("Doubly Linked List is empty. Create a list first.\n");
            } else {
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                printf("Enter the target value: ");
                scanf("%d", &targetValue);
                insertLeft(&head, value, targetValue);
            }
            break;

        case 3:
            if (head == NULL) {
                printf("Doubly Linked List is empty. Nothing to delete.\n");
            } else {
                printf("Enter the value to delete: ");
                scanf("%d", &value);
                deleteNode(&head, value);
            }
            break;

        case 4:
            if (head == NULL) {
                printf("Doubly Linked List is empty.\n");
            } else {
                displayList(head);
            }
            break;

        case 5:
            printf("Exiting program.\n");
            break;

        default:
            printf("Invalid choice. Please enter a valid option.\n");
    }
} while (choice != 5);
```

```
    return 0;
}
```

OUTPUT