WAP to Implement doubly link list with primitive operations

a)
Create a doubly linked list.

b)
Insert a new node to the left of the node.

c)
Delete the node based on a specific value

Display the contents of the list

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

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

struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertToLeft(struct Node** head, int value) {
    struct Node* newNode = createNode(value);
    if (*head == NULL) {
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
```

```
        *head = newNode;
    }
}

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

    struct Node* current = *head;

    while (current != NULL) {
        if (current->data == value) {
            if (current->prev != NULL) {
                current->prev->next = current->next;
            } else {
                *head = 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);
}

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

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

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

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

    do {
        printf("\n1. Insert to the left\n2. Delete by value\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                insertToLeft(&head, value);
                break;
            case 2:
                printf("Enter the value to delete: ");
                scanf("%d", &value);
                deleteNode(&head, value);
                break;
            case 3:
                displayList(head);
                break;
            case 4:
                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }
    } while (choice != 4);

    return 0;
}
```

```
1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 12

1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 13

1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 14

1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 15

1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 3
Doubly Linked List: 15 <-> 14 <-> 13 <-> 12 <-> NULL
```

```
1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 2
Enter the value to delete: 13
Node with value 13 deleted.

1. Insert to the left
2. Delete by value
3. Display
4. Exit
Enter your choice: 3
Doubly Linked List: 15 <-> 14 <-> 12 <-> NULL
```