7. 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

d) Display the contents of the list

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

// Structure of doubly linked list node
struct Node {
    int data;
    struct Node *prev;
    struct Node *next;
};

// Create doubly linked list
struct Node* createList() {
    struct Node *head = NULL, *temp, *newNode;
    int n, value;

    printf("Enter number of nodes: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        printf("Enter data: ");
        scanf("%d", &value);

        newNode->data = value;
        newNode->prev = newNode->next = NULL;

        if (head == NULL) {
```

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

// Insert a new node to the left of a given value
struct Node* insertLeft(struct Node *head, int key, int value) {
    struct Node *temp = head;

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

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

    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);
    return head;
}

// Delete node with specific value
struct Node* deleteNode(struct Node *head, int key) {
    struct Node *temp = head;

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

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

    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    else
        head = temp->next;

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

    free(temp);
    printf("Deleted %d from list\n", key);
    return head;
```

```
}

// Display the list
void display(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("\n");
}

// Main function
int main() {
    struct Node *head = NULL;
    int choice, key, value;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Create List\n");
        printf("2. Insert Left of Node\n");
        printf("3. Delete Node\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
```

```
        switch (choice) {
            case 1:
                head = createList();
                break;
            case 2:
                printf("Enter existing value and new value: ");
                scanf("%d %d", &key, &value);
                head = insertLeft(head, key, value);
                break;
            case 3:
                printf("Enter value to delete: ");
                scanf("%d", &key);
                head = deleteNode(head, key);
                break;
            case 4:
                display(head);
                break;
            case 5:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
}
```

```
Menu:
1. Create List
2. Insert Left of Node
3. Delete Node
4. Display List
5. Exit
Enter choice: 1
Enter number of nodes: 4
Enter data: 10
Enter data: 20
Enter data: 30
Enter data: 40

Menu:
1. Create List
2. Insert Left of Node
3. Delete Node
4. Display List
5. Exit
Enter choice: 2
Enter existing value and new value: 50

5
Value 50 not found

Menu:
1. Create List
2. Insert Left of Node
3. Delete Node
4. Display List
5. Exit
Enter choice: 3
Enter value to delete: 50
Value 50 not found

Menu:
1. Create List
2. Insert Left of Node
3. Delete Node
4. Display List
5. Exit
Enter choice: 4
Doubly Linked List: 10 20 30 40

Menu:
1. Create List
2. Insert Left of Node
3. Delete Node
4. Display List
5. Exit
Enter choice: 5

Process returned 0 (0x0)   execution time : 35.554 s
Press any key to continue.
```