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>
struct node {
    int data;
    struct node *prev;
    struct node *next;
};
struct node *head = NULL;
void create() {
    int n, i, val;
    struct node *newnode, *temp;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Enter data: ");
        scanf("%d", &val);
        newnode->data = val;
        newnode->prev = NULL;
        newnode->next = NULL;
        if (head == NULL) {
            head = newnode;
            temp = head;
        } else {
            temp->next = newnode;
            newnode->prev = temp;
```

```
            temp = newnode;
        }
    }
}
void insert_left() {
    int key, val;
    struct node *newnode, *temp;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("Enter value to insert left of: ");
    scanf("%d", &key);
    printf("Enter new data: ");
    scanf("%d", &val);
    temp = head;
    while (temp != NULL && temp->data != key) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Value not found\n");
        return;
    }
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = val;
    newnode->next = temp;
    newnode->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = newnode;
    else
        head = newnode;
    temp->prev = newnode;
```

```
    printf("Node inserted successfully\n");
}
void delete_value() {
    int key;
    struct node *temp;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("Enter value to delete: ");
    scanf("%d", &key);
    temp = head;
    while (temp != NULL && temp->data != key) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Value not found\n");
        return;
    }
    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("Node deleted successfully\n");
}
void display() {
    struct node *temp;
    if (head == NULL) {
```

```
        printf("List is empty\n");
        return;
    }
    temp = head;
    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    int choice;
    do {
        printf("\n--- DOUBLY LINKED LIST MENU ---\n");
        printf("1. Create\n");
        printf("2. Insert Left of a Node\n");
        printf("3. Delete by Value\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: create(); break;
            case 2: insert_left(); break;
            case 3: delete_value(); break;
            case 4: display(); break;
            case 5: exit(0);
            default: printf("Invalid choice\n");
        }
    } while (1);
    return 0;}
```

```
--- DOUBLY LINKED LIST MENU ---
1. Create
2. Insert Left of a Node
3. Delete by Value
4. Display
5. Exit
Enter your choice: 1
Enter number of nodes: 3
Enter data: 10
Enter data: 20
Enter data: 30

--- DOUBLY LINKED LIST MENU ---
1. Create
2. Insert Left of a Node
3. Delete by Value
4. Display
5. Exit
Enter your choice: 2
Enter value to insert left of: 4
Enter new data: 40
Value not found

--- DOUBLY LINKED LIST MENU ---
1. Create
2. Insert Left of a Node
3. Delete by Value
4. Display
5. Exit
Enter your choice: 2
Enter value to insert left of: 20
Enter new data: 50
Node inserted successfully
```

```
--- DOUBLY LINKED LIST MENU ---
1. Create
2. Insert Left of a Node
3. Delete by Value
4. Display
5. Exit
Enter your choice: 3
Enter value to delete: 20
Node deleted successfully

--- DOUBLY LINKED LIST MENU ---
1. Create
2. Insert Left of a Node
3. Delete by Value
4. Display
5. Exit
Enter your choice: 4
Doubly Linked List: 10 <-> 50 <-> 30 <-> NULL

--- DOUBLY LINKED LIST MENU ---
1. Create
2. Insert Left of a Node
3. Delete by Value
4. Display
5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 60.698 s
Press any key to continue.
```