```
---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 4
DLL: 10 <-> 20 <-> 30 <-> NULL

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 3
Enter value to delete: 15
Value 15 not found in list

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 4
DLL: 10 <-> 20 <-> 30 <-> NULL

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 38.005 s
Press any key to continue.
```

```
---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 1
Enter value to insert: 10

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 1
Enter value to insert: 20

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 1
Enter value to insert: 30

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 2
Insert new value: 20
Insert to the left of value: 15
Node with value 15 not found

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
3. Delete a node by value
4. Display list
5. Exit
Enter your choice: 4
DLL: 10 <-> 20 <-> 30 <-> NULL

---- Doubly Linked List Menu ----
1. Create (Insert at end)
2. Insert to left of a node
```

```c
            if (temp == head) {
                head = temp->next;
                if (head != NULL)
                    head->prev = NULL;
                free(temp);
                return;
            }

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

            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
            free(temp);
    }

    void display() {
        struct node *temp = head;

        printf("DLL: ");
        while (temp != NULL) {
            printf("%d <-> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }

    int main() {
        int choice, value, x, pos;

        while (1) {
            printf("\n---- Doubly Linked List Menu ----\n");
            printf("1. Create (Insert at end)\n");
            printf("2. Insert to left of a node\n");
```

```c
        printf("2. Insert to left of a node\n");
        printf("3. Delete a node by value\n");
        printf("4. Display list\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            printf("Enter value to insert: ");
            scanf("%d", &value);
            insertAtEnd(value);
            break;

        case 2:
            printf("Insert new value: ");
            scanf("%d", &value);
            printf("Insert to the left of value: ");
            scanf("%d", &x);
            insertLeft(value, x);
            break;

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

        case 4:
            display();
            break;

        case 5:
            exit(0);

        default:
            printf("Invalid choice! Try again.\n");
```

```c
            scanf("%d", &choice);

            switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                insertAtEnd(value);
                break;

            case 2:
                printf("Insert new value: ");
                scanf("%d", &value);
                printf("Insert to the left of value: ");
                scanf("%d", &x);
                insertLeft(value, x);
                break;

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

            case 4:
                display();
                break;

            case 5:
                exit(0);

            default:
                printf("Invalid choice! Try again.\n");
            }
        }
        return 0;
    }
```

```c
        struct node *temp = head;

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

        if (temp == NULL) {
            printf("Node with value %d not found\n", existingValue);
            return;
        }

        struct node *newNode = createNode(newValue);

        if (temp == head) {
            newNode->next = head;
            head->prev = newNode;
            head = newNode;
            return;
        }

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

void deleteByValue(int value) {
        struct node *temp = head;

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

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

        if (temp == head) {
```

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

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

struct node *head = NULL;

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;
}

void insertAtEnd(int value) {
    struct node *newNode = createNode(value);

    if (head == NULL) {
        head = newNode;
        return;
    }

    struct node *temp = head;
    while (temp->next != NULL)
        temp = temp->next;

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

void insertLeft(int newValue, int existingValue) {
    struct node *temp = head;
```