```
#include <stdio.h>
 #include <stdlib.h>
typedef struct Node {
 int data;
struct Node* prev;
 struct Node* next; } Node;
Node* createNode(int data) {
Node* newNode = (Node*)malloc(sizeof(Node));
newNode->data = data;
newNode->prev = NULL;
newNode->next = NULL;
return newNode; }
Node* createDoublyLinkedList(int n) {
 Node* head = NULL;
 Node* tail = NULL;
 int value;
 for (int i = 0; i < n; i++) {
printf("Enter value for node %d: ", i + 1);
 scanf("%d", &value);
Node* newNode = createNode(value);
 if (head == NULL) {
head = newNode;
```

```
tail = newNode; }
else { tail->next = newNode;
newNode->prev = tail;
 tail = newNode; } }
 return head; }
Node* insertLeft(Node* head, int target, int value) {
 Node* current = head;
 while (current != NULL && current->data != target) {
current = current->next; }
if (current == NULL) {
 printf("Target node not found.\n");
return head; }
 Node* newNode = createNode(value);
newNode->next = current;
newNode->prev = current->prev;
 if (current->prev != NULL) {
 current->prev->next = newNode; }
 else { head = newNode;
} current->prev = newNode;
 return head; }
Node* deleteNode(Node* head, int value) {
Node* current = head;
```

```
 while (current != NULL && current->data != value) {
current = current->next; }
 if (current == NULL) {
printf("Node with value %d not found.\n", value);
return head; }
 if (current->prev != NULL) {
 current->prev->next = current->next; }
else
 { head = current->next; }
 if (current->next != NULL) {
 current->next->prev = current->prev; }
 free(current);
return head; }
void displayList(Node* head) {
Node* temp = head;
printf("Doubly Linked List: ");
 while (temp != NULL) {
printf("%d <-> ", temp->data);
 temp = temp->next; }
 printf("NULL\n"); }
int main() {
 Node* list = NULL;
```

int n, choice, target, value;

```
printf("Enter number of nodes for the doubly linked list: ");
scanf("%d", &n);
list = createDoublyLinkedList(n);
displayList(list);

while (1) {
    printf("\nMenu:\n");
    printf("1. Insert to the left of a node\n");
    printf("2. Delete a node by value\n");
    printf("3. Display the list\n");
    printf("4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter the target node value: ");
            scanf("%d", &target);
            printf("Enter the value to insert: ");
            scanf("%d", &value);
            list = insertLeft(list, target, value);
            displayList(list);
            break;
        case 2:
            printf("Enter the value of the node to delete: ");
            scanf("%d", &value);
            list = deleteNode(list, value);
            displayList(list);
            break;
        case 3:
            displayList(list);
            break;
        case 4:
            printf("Exiting program.\n");
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
```

```
    }
}

return 0;


}
```

```
Enter number of nodes for the doubly linked list: 3
Enter value for node 1: 10
Enter value for node 2: 20
Enter value for node 3: 30
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL

Menu:
1. Insert to the left of a node
2. Delete a node by value
3. Display the list
4. Exit
Enter your choice: 1
Enter the target node value: 20
Enter the value to insert: 15
Doubly Linked List: 10 <-> 15 <-> 20 <-> 30 <-> NULL
```

```
Menu:
1. Insert to the left of a node
2. Delete a node by value
3. Display the list
4. Exit
Enter your choice: 2
Enter the value of the node to delete: 15
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL

Menu:
1. Insert to the left of a node
2. Delete a node by value
3. Display the list
4. Exit
Enter your choice: 4
Exiting program.
```