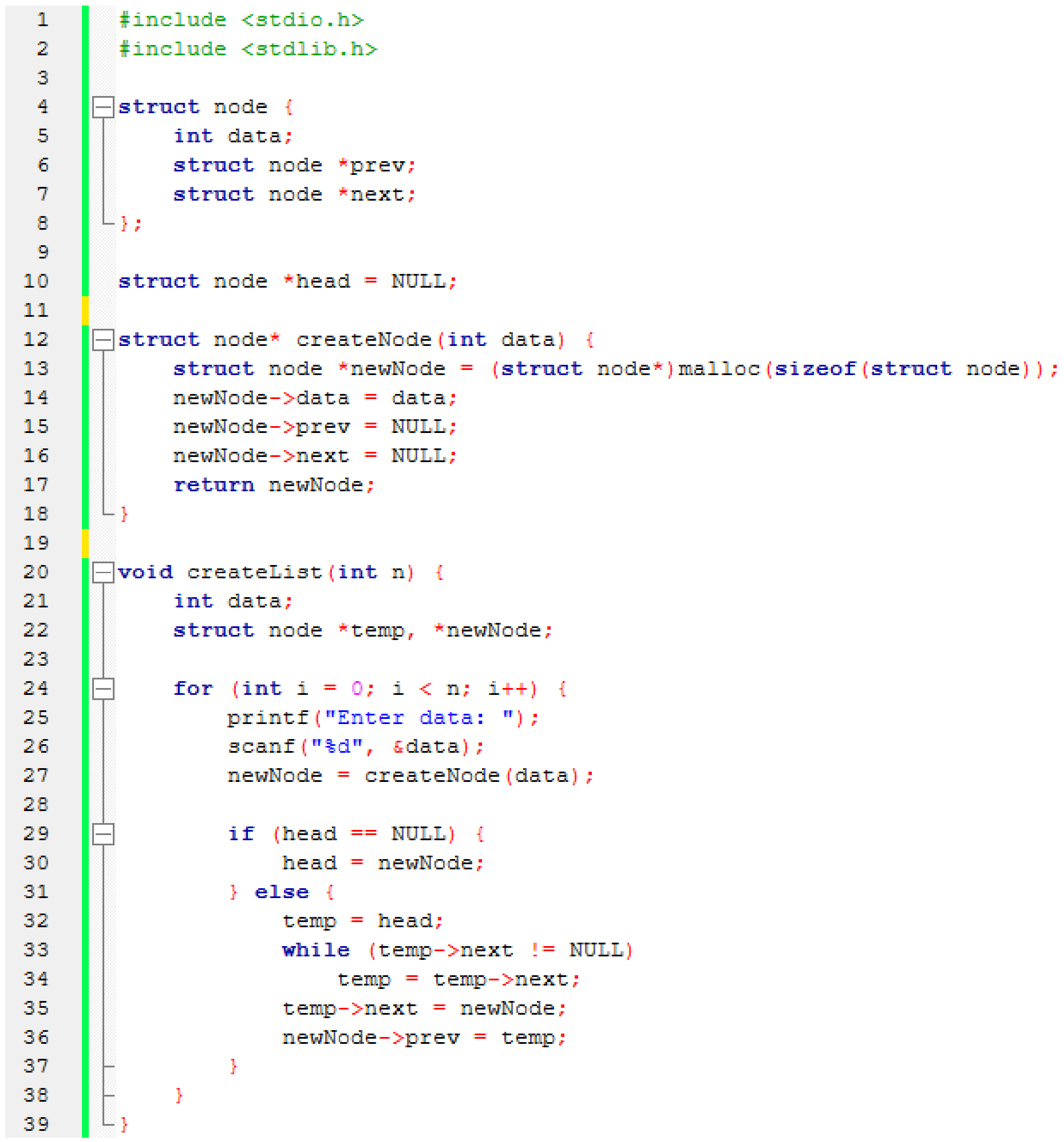
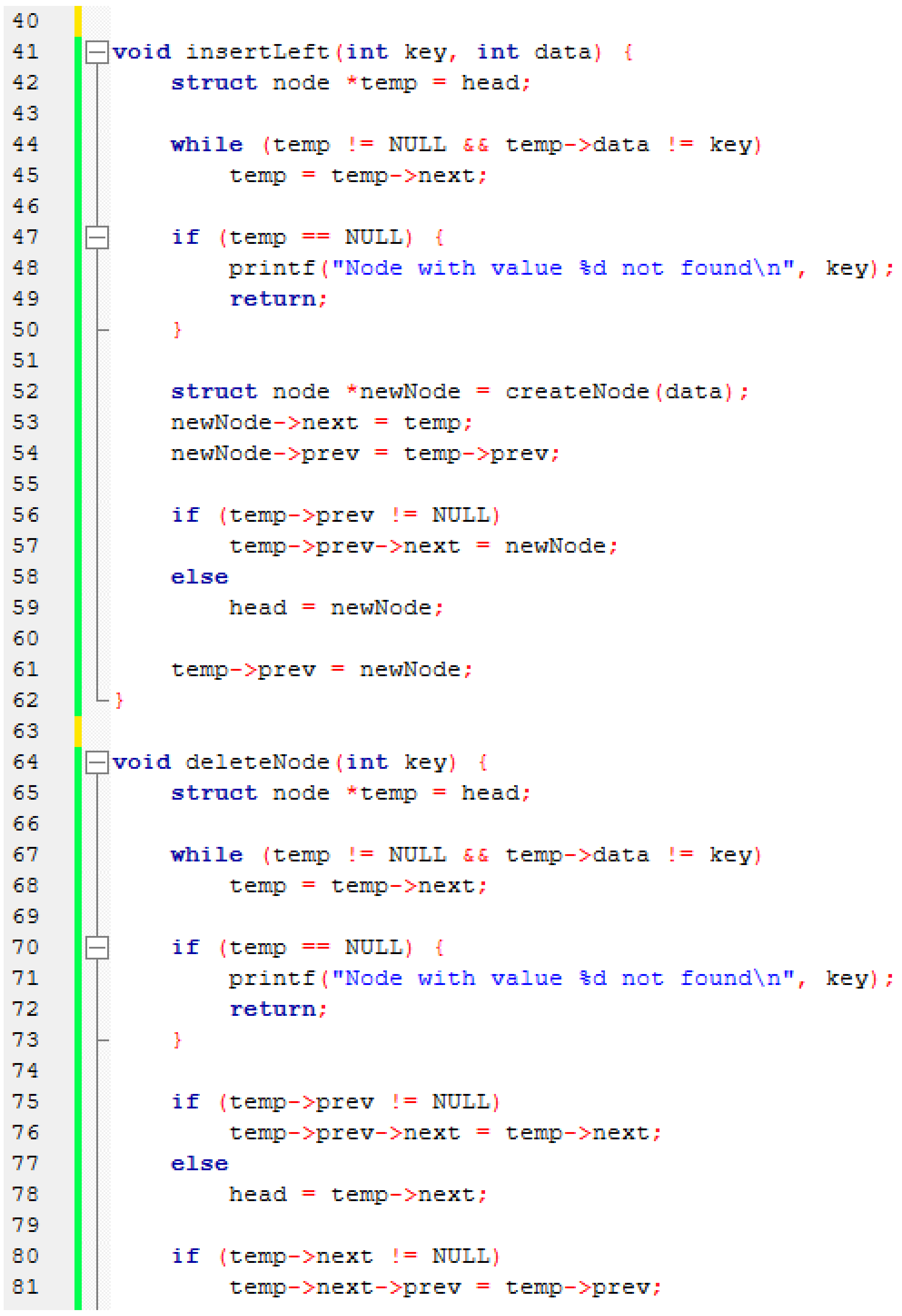
Doubly-LinkedList(LAB-7) [29/11/25]

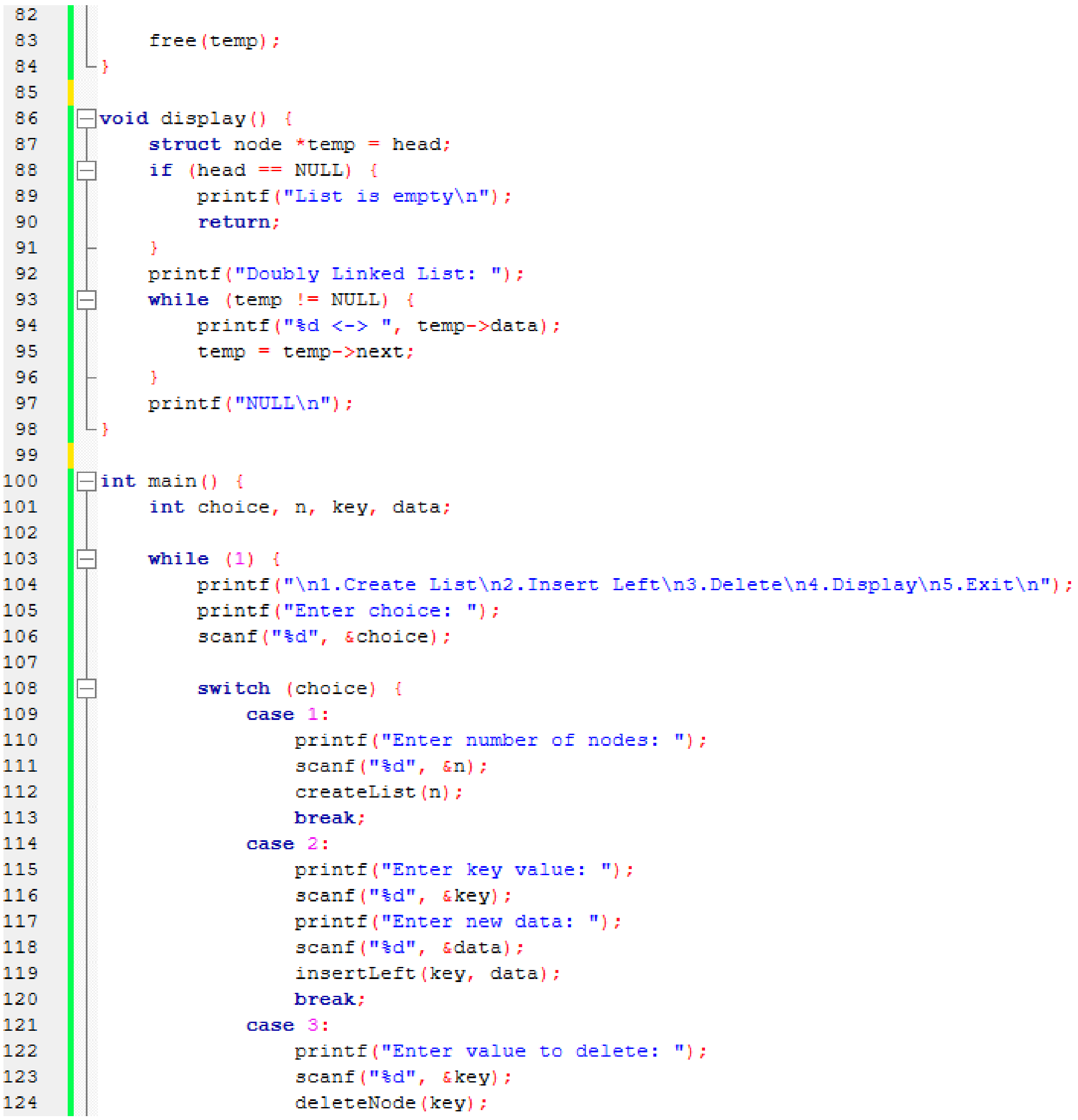
code:



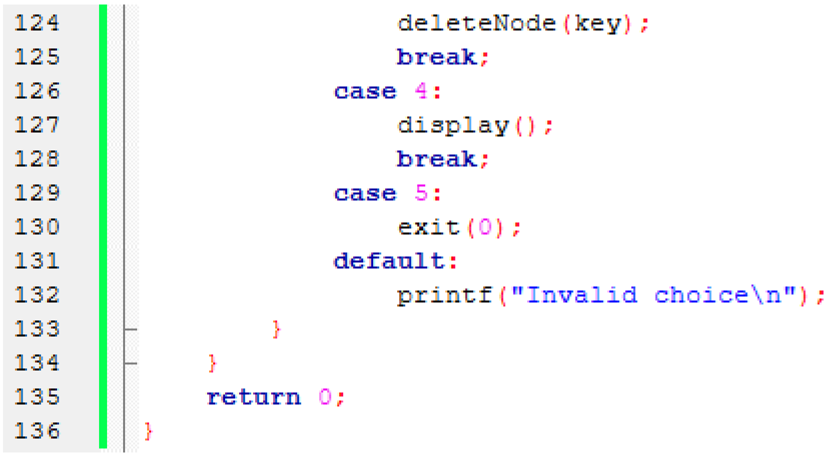
```c
#include <stdio.h>
#include <stdlib.h>

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

struct node *head = NULL;

struct node* createNode(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void createList(int n) {
    int data;
    struct node *temp, *newNode;

    for (int i = 0; i < n; i++) {
        printf("Enter data: ");
        scanf("%d", &data);
        newNode = createNode(data);

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



```c

void insertLeft(int key, int data) {
    struct node *temp = head;

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

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

    struct node *newNode = createNode(data);
    newNode->next = temp;
    newNode->prev = temp->prev;

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

    temp->prev = newNode;
}

void deleteNode(int key) {
    struct node *temp = head;

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

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

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

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

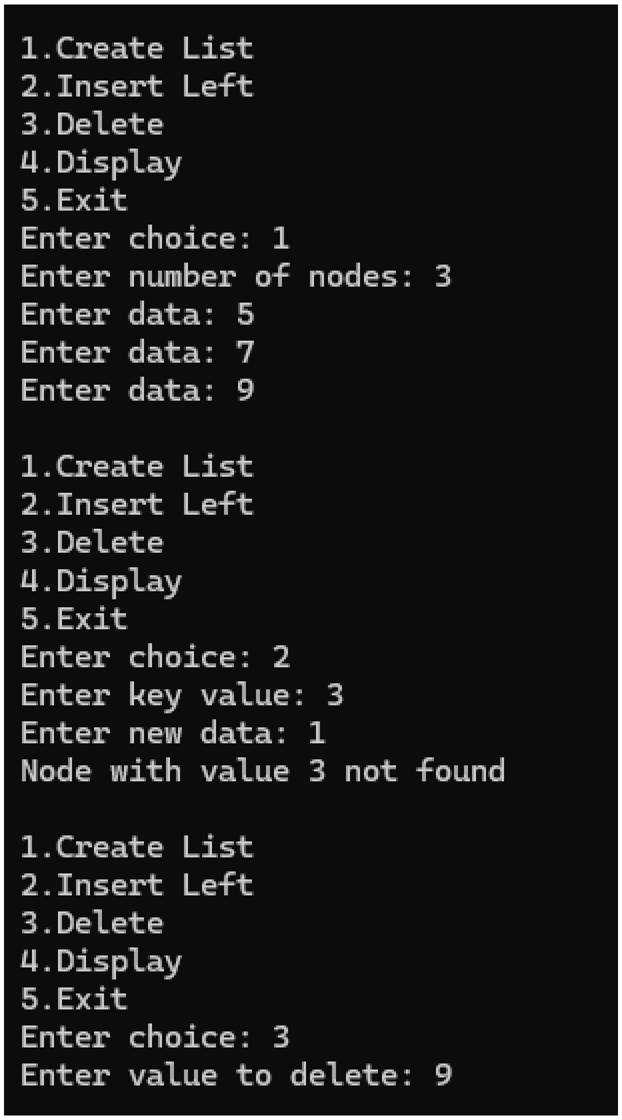


```c

    free(temp);
}

void display() {
    struct node *temp = head;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, n, key, data;

    while (1) {
        printf("\n1.Create List\n2.Insert Left\n3.Delete\n4.Display\n5.Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter key value: ");
                scanf("%d", &key);
                printf("Enter new data: ");
                scanf("%d", &data);
                insertLeft(key, data);
                break;
            case 3:
                printf("Enter value to delete: ");
                scanf("%d", &key);
                deleteNode(key);
```

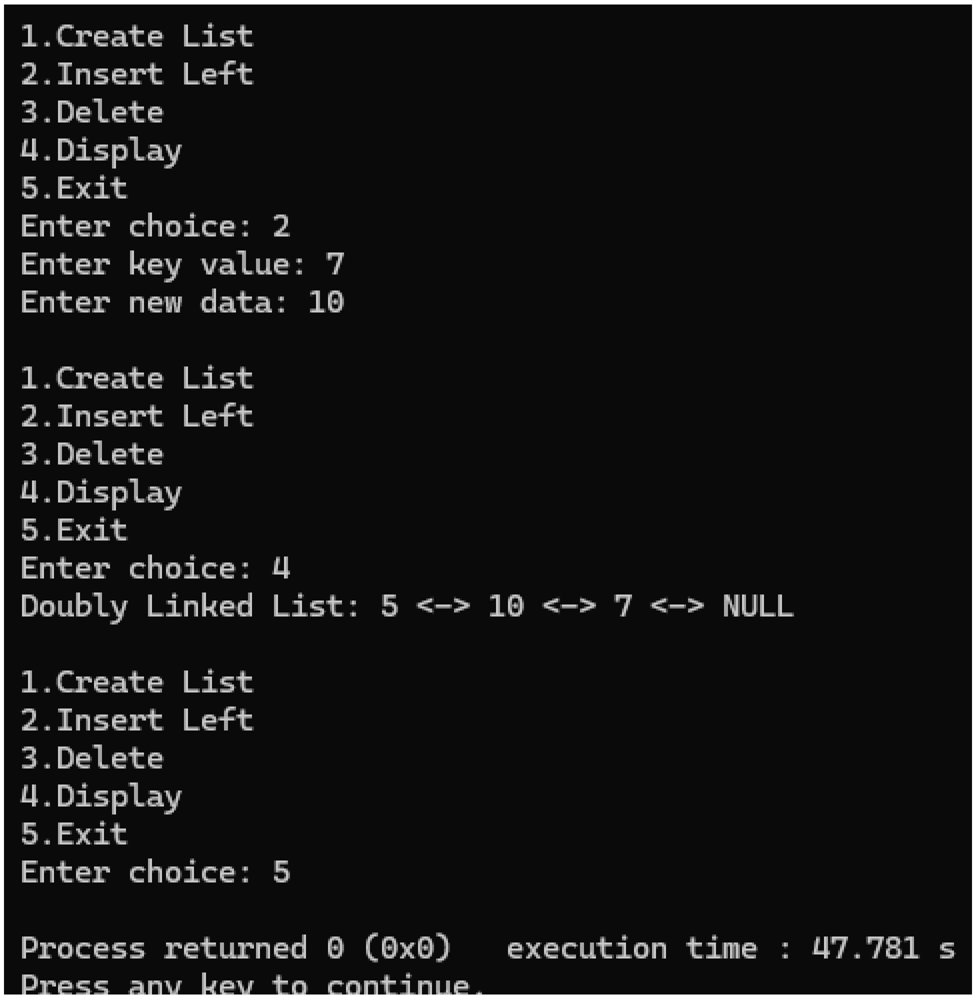


```c
                deleteNode(key);
                break;
            case 4:
                display();
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Output:



```
1.Create List
2.Insert Left
3.Delete
4.Display
5.Exit
Enter choice: 1
Enter number of nodes: 3
Enter data: 5
Enter data: 7
Enter data: 9

1.Create List
2.Insert Left
3.Delete
4.Display
5.Exit
Enter choice: 2
Enter key value: 3
Enter new data: 1
Node with value 3 not found

1.Create List
2.Insert Left
3.Delete
4.Display
5.Exit
Enter choice: 3
Enter value to delete: 9
```



```
1.Create List
2.Insert Left
3.Delete
4.Display
5.Exit
Enter choice: 2
Enter key value: 7
Enter new data: 10

1.Create List
2.Insert Left
3.Delete
4.Display
5.Exit
Enter choice: 4
Doubly Linked List: 5 <-> 10 <-> 7 <-> NULL

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

Process returned 0 (0x0)   execution time : 47.781 s
Press any key to continue.
```