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>

/* Definition of Doubly Linked List Node */
struct node {
   int data;
   struct node *prev;
   struct node *next;
};
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;
}

/* Create Doubly Linked List */
struct node* createList() {
   struct node *head = NULL, *temp = NULL, *newNode;
   int n, data, i;

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

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

```
        newNode = createNode(data);

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

/* Insert a node to the left of a given value */
struct node* insertLeft(struct node *head, 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 head;
    }

    struct node *newNode = createNode(data);

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

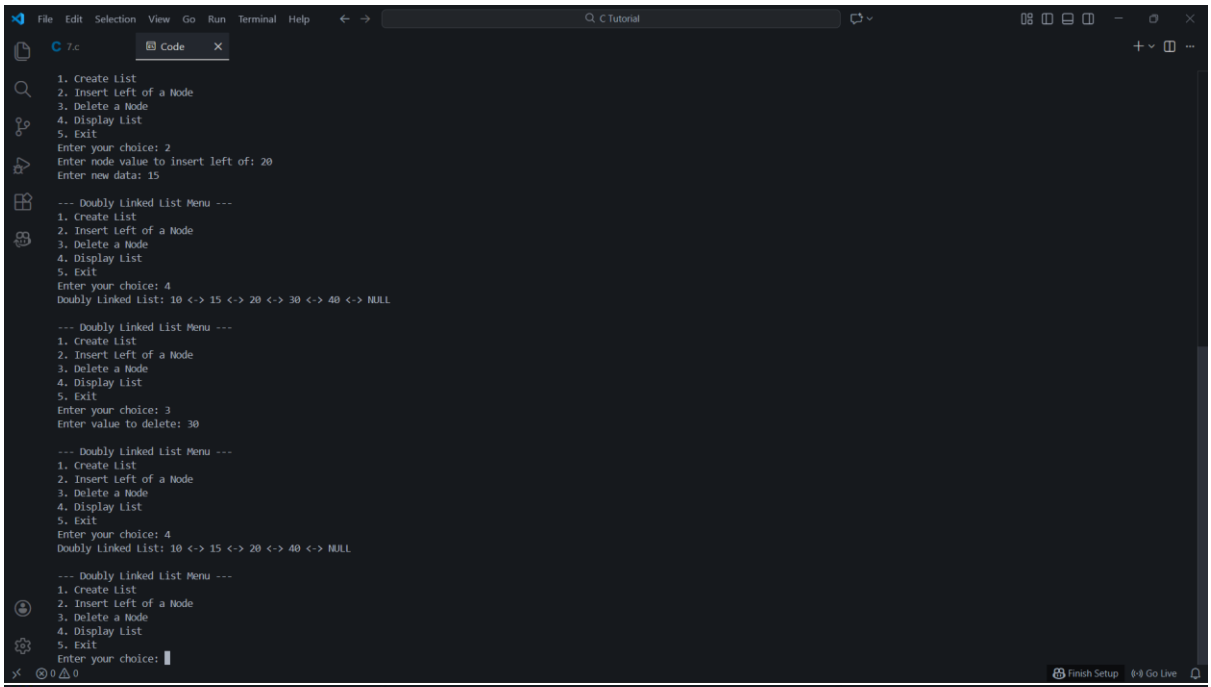
```
    if (temp->prev != NULL)
        temp->prev->next = newNode;
    else
        head = newNode;  // Insert at beginning

    temp->prev = newNode;

    return head;
}

/* Delete a node based on 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("Node with value %d not found\n", key);
        return head;
    }

    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    else
        head = temp->next;  // Delete first node

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

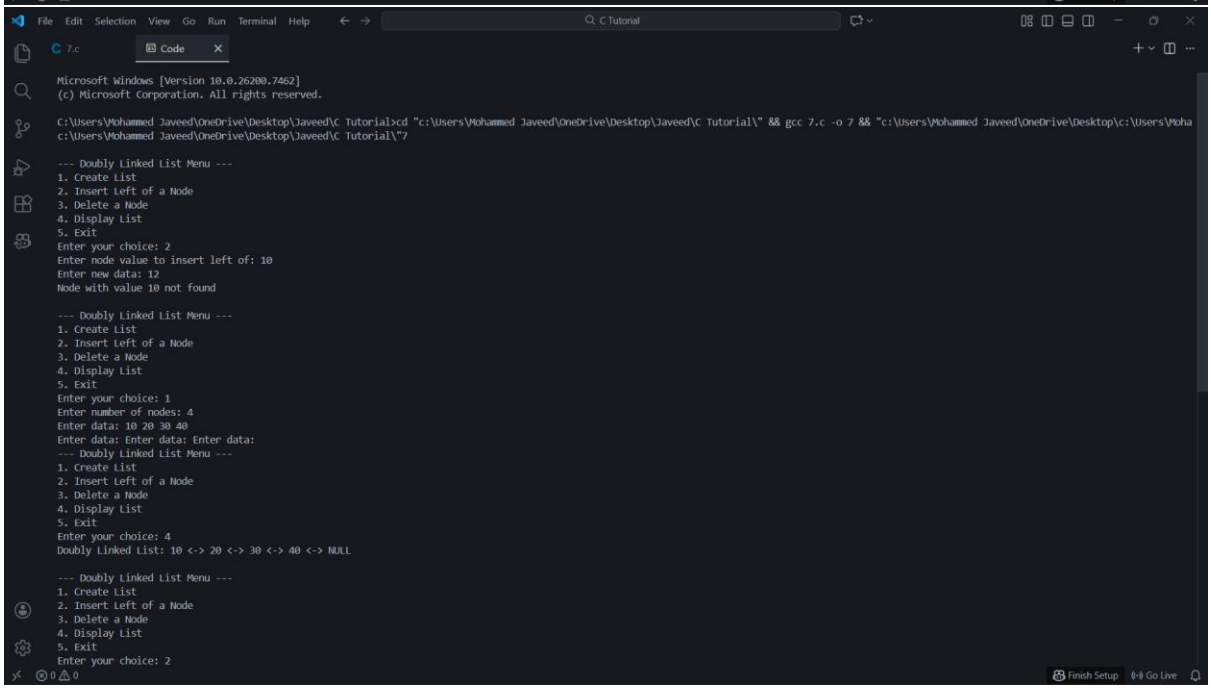
```
    free(temp);
    return head;
}

/* Display the list */
void display(struct node *head) {
    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");
}

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

    do {
        printf("\n--- Doubly Linked List Menu---\n");
        printf("1. Create List\n");
        printf("2. Insert Left of a Node\n");
        printf("3. Delete a Node\n");
```

```
printf("4. Display List\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
    case 1:
        head = createList();
        break;

    case 2:
        printf("Enter node value to insert left of: ");
        scanf("%d", &key);
        printf("Enter new data: ");
        scanf("%d", &data);
        head = insertLeft(head, key, data);
        break;

    case 3:
        printf("Enter value to delete: ");
        scanf("%d", &key);
        head = deleteNode(head, key);
        break;

    case 4:
        display(head);
        break;

    case 5:
        printf("Exiting program\n");
        break;
```

```
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 5);

    return 0;
}
```

OUTPUT:


1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 2
Enter node value to insert left of: 20
Enter new data: 15
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 4
Doubly Linked List: 10 <-> 15 <-> 20 <-> 30 <-> 40 <-> NULL
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 3
Enter value to delete: 30
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 4
Doubly Linked List: 10 <-> 15 <-> 20 <-> 40 <-> NULL
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice:


Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\" && gcc 7.c -o 7 && "c:\Users\Mohammed Javeed\OneDrive\Desktop\c:\Users\Moha
c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\C Tutorial\"7
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 2
Enter node value to insert left of: 10
Enter new data: 12
Node with value 10 not found
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 1
Enter number of nodes: 4
Enter data: 10 20 30 40
Enter data: Enter data: Enter data:
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 4
Doubly Linked List: 10 <-> 20 <-> 30 <-> 40 <-> NULL
--- Doubly Linked List Menu ---
1. Create List
2. Insert Left of a Node
3. Delete a Node
4. Display List
5. Exit
Enter your choice: 2