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

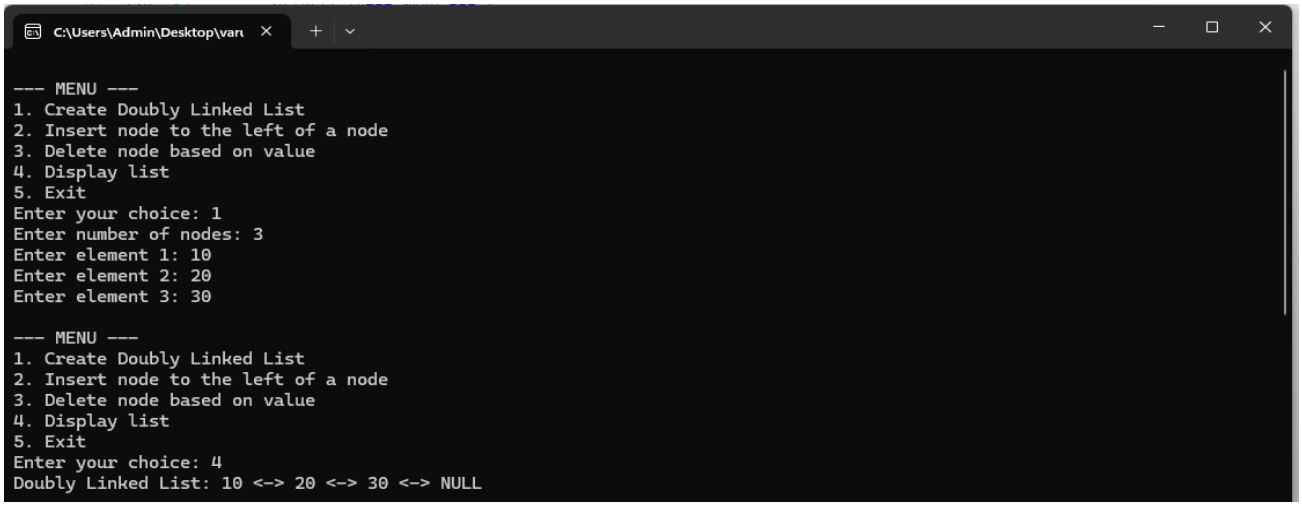
```
#include <stdio.h>
#include <stdlib.h>

/* Node structure */
struct node {
   int data;
   struct node *prev;
   struct node *next;
};

/* Create a new node */
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, int n) {
   int data;
   struct node *temp, *newNode;

   for (int i = 0; i < n; i++) {
      printf("Enter element %d: ", i + 1);
      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;
      }
   }
   return head;
}
```

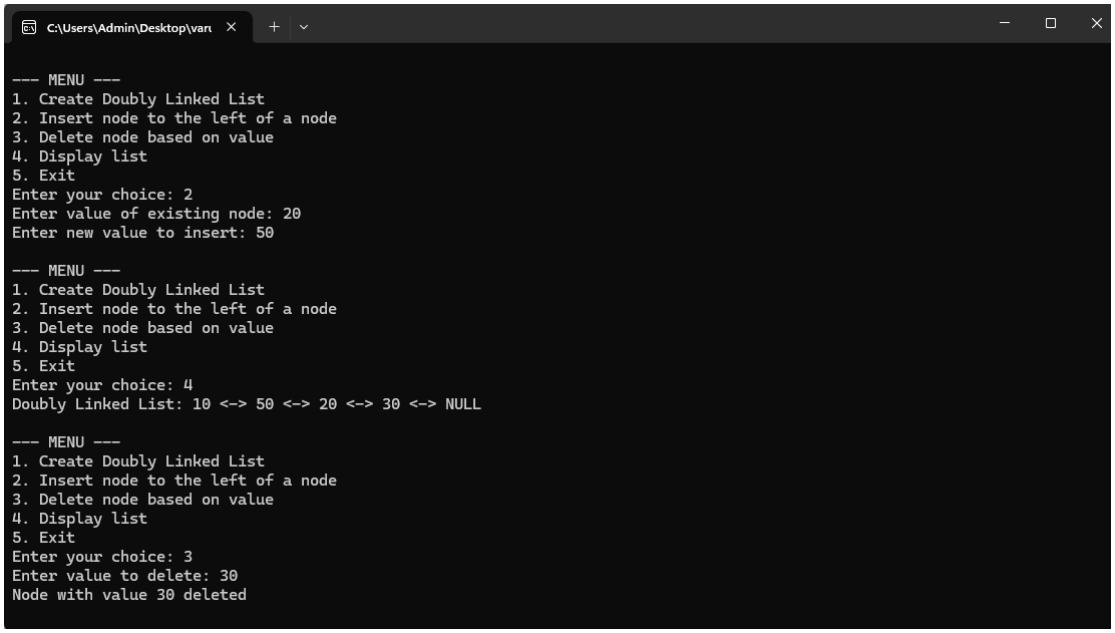
```
/* Insert node to the left of a given value */
struct node* insertLeft(struct node *head, int key, int data) {
   struct node *temp = head;

   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;

   temp->prev = newNode;

   return head;
}

/* Delete 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;

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

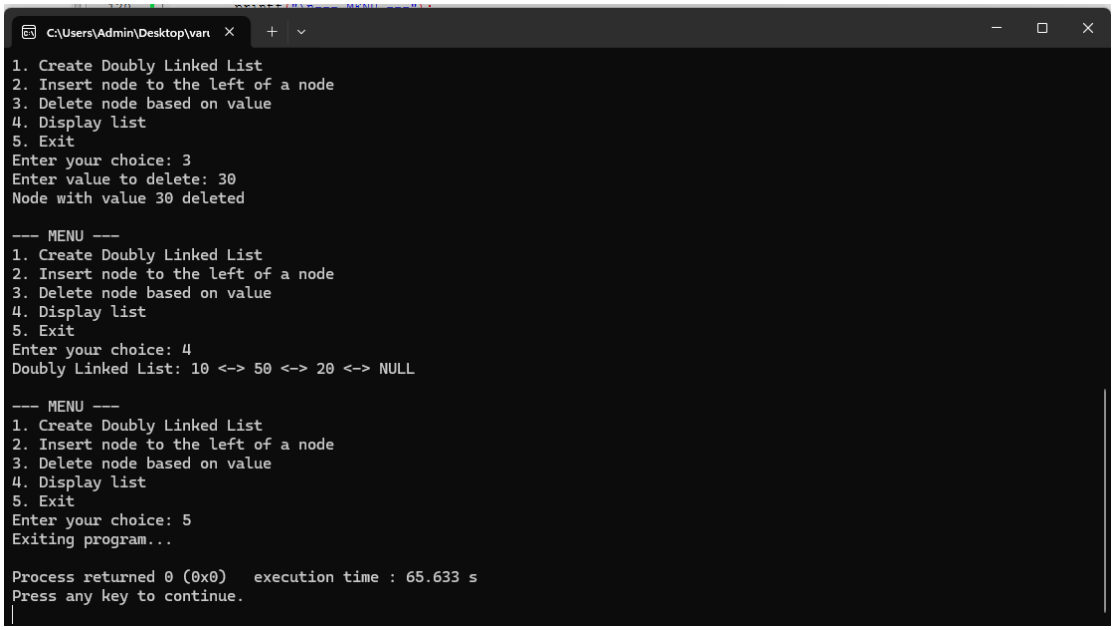
```
    free(temp);
    printf("Node with value %d deleted\n", key);

    return head;
}

/* Display 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, n, key, data;

    do {
        printf("\n--- MENU ---");
        printf("\n1. Create Doubly Linked List");
        printf("\n2. Insert node to the left of a node");
        printf("\n3. Delete node based on value");
        printf("\n4. Display list");
        printf("\n5. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                head = NULL;
                head = createList(head, n);
                break;
```

```
            case 2:
                printf("Enter value of existing node: ");
                scanf("%d", &key);
                printf("Enter new value to insert: ");
                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");
                exit(0);

            default:
                printf("Invalid choice\n");
        }
    } while (1);

    return 0;
}
```

Output:




C:\Users\Admin\Desktop\var
--- MENU ---
1. Create Doubly Linked List
2. Insert node to the left of a node
3. Delete node based on value
4. Display list
5. Exit
Enter your choice: 2
Enter value of existing node: 20
Enter new value to insert: 50
--- MENU ---
1. Create Doubly Linked List
2. Insert node to the left of a node
3. Delete node based on value
4. Display list
5. Exit
Enter your choice: 4
Doubly Linked List: 10 <-> 50 <-> 20 <-> 30 <-> NULL
--- MENU ---
1. Create Doubly Linked List
2. Insert node to the left of a node
3. Delete node based on value
4. Display list
5. Exit
Enter your choice: 3
Enter value to delete: 30
Node with value 30 deleted


C:\Users\Admin\Desktop\var
1. Create Doubly Linked List
2. Insert node to the left of a node
3. Delete node based on value
4. Display list
5. Exit
Enter your choice: 3
Enter value to delete: 30
Node with value 30 deleted
--- MENU ---
1. Create Doubly Linked List
2. Insert node to the left of a node
3. Delete node based on value
4. Display list
5. Exit
Enter your choice: 4
Doubly Linked List: 10 <-> 50 <-> 20 <-> NULL
--- MENU ---
1. Create Doubly Linked List
2. Insert node to the left of a node
3. Delete node based on value
4. Display list
5. Exit
Enter your choice: 5
Exiting program...
Process returned 0 (0x0)   execution time : 65.633 s
Press any key to continue.