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

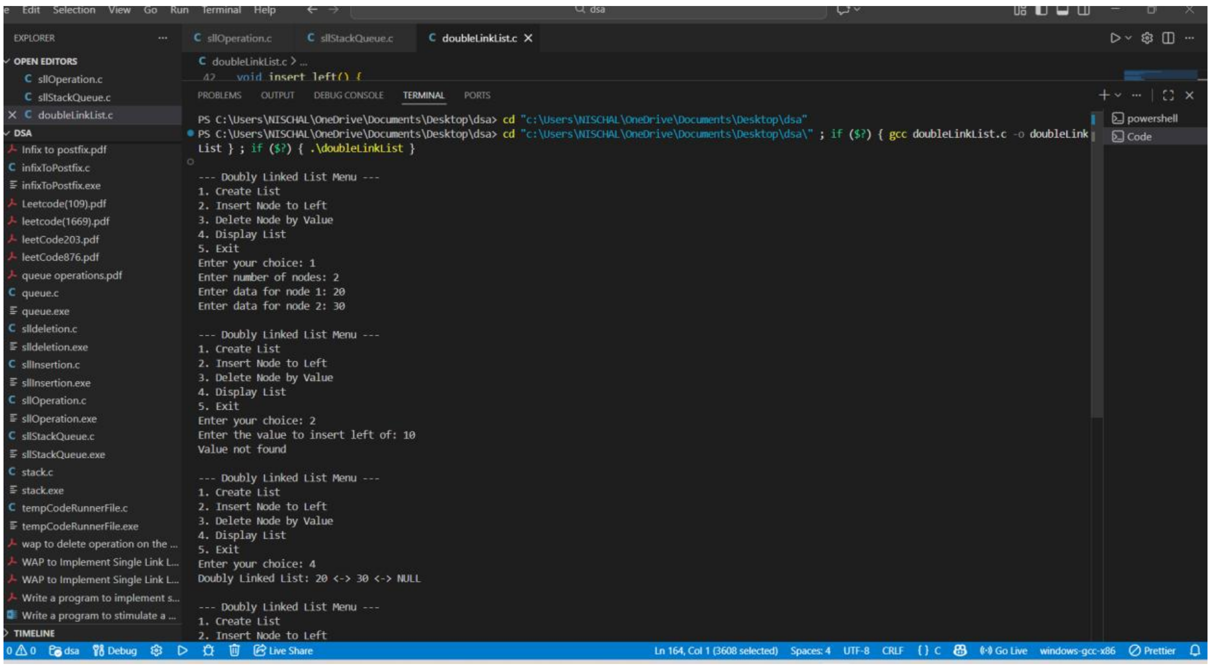
```
#include <stdio.h>

#include <stdlib.h>


// Structure definition

struct node {

  int data;

  struct node *prev;

  struct node *next;

};


struct node *head = NULL;


// Function to create a doubly linked list

void create() {

  int n, i, value;

  struct node *temp, *newnode;


  printf("Enter number of nodes: ");

  scanf("%d", &n);


  for (i = 0; i < n; i++) {

    newnode = (struct node *)malloc(sizeof(struct node));

    printf("Enter data for node %d: ", i + 1);

    scanf("%d", &value);


    newnode->data = value;
    newnode->prev = NULL;
    newnode->next = NULL;

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

// Function to insert a node to the left of a given value
void insert_left() {
  int key, value;
  struct node *temp, *newnode;

  if (head == NULL) {
    printf("List is empty\n");
    return;
  }

  printf("Enter the value to insert left of: ");
  scanf("%d", &key);

  temp = head;
```

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

    if (temp == NULL) {
        printf("Value not found\n");
        return;
    }

    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter new data: ");
    scanf("%d", &value);

    newnode->data = value;
    newnode->next = temp;
    newnode->prev = temp->prev;

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

    temp->prev = newnode;
}

// Function to delete a node with a specific value
void delete_value() {
    int key;
    struct node *temp;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

    printf("Enter value to delete: ");
    scanf("%d", &key);

    temp = head;

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

    if (temp == NULL) {
        printf("Value not found\n");
        return;
    }

    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 deleted successfully\n");
}
```

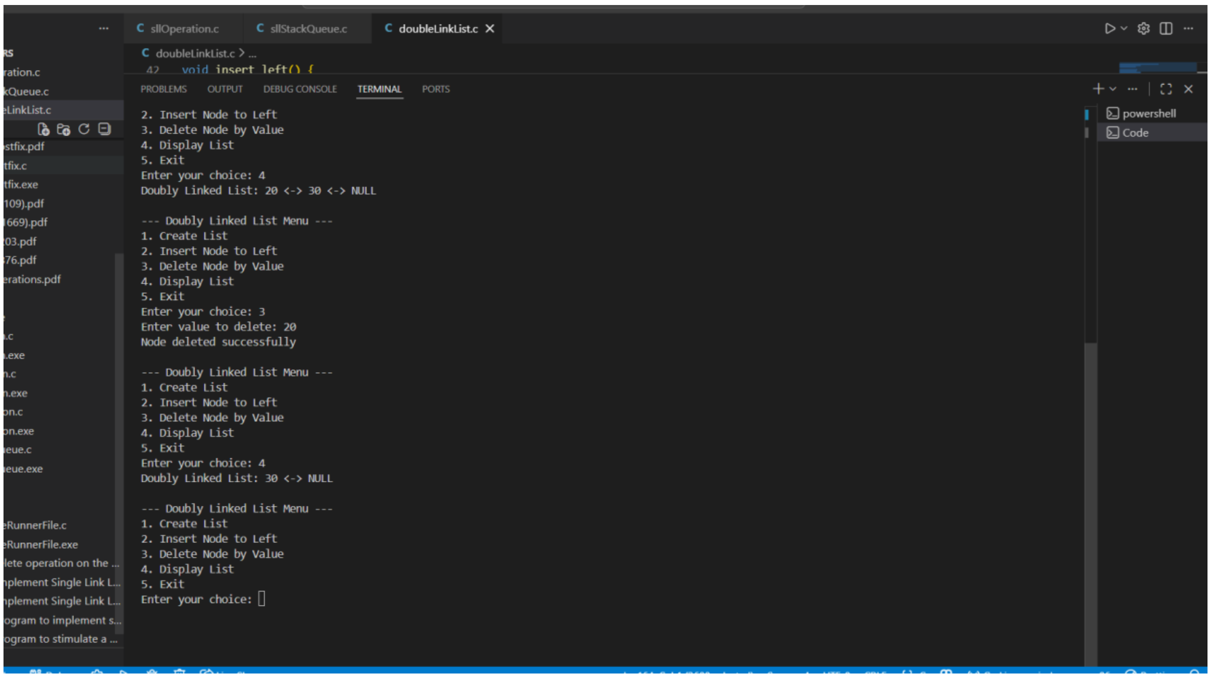
```
// Function to display the list
void display() {
  struct node *temp;

  if (head == NULL) {
    printf("List is empty\n");
    return;
  }

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

// Main function (Menu Driven)
int main() {
  int choice;

  do {
    printf("\n--- Doubly Linked List Menu ---\n");
    printf("1. Create List\n");
    printf("2. Insert Node to Left\n");
    printf("3. Delete Node by Value\n");
    printf("4. Display List\n");
```

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

        switch (choice) {
          case 1: create();
                break;
          case 2: insert_left();
                break;
          case 3: delete_value();
                break;
          case 4: display();
                break;
          case 5: printf("Exiting program\n");
                break;
          default: printf("Invalid choice\n");
        }
    } while (choice != 5);

    return 0;
}
```

OUTPUT:



```
PS C:\Users\NISCHAL\OneDrive\Documents\Desktop\dsa> cd "c:\Users\NISCHAL\OneDrive\Documents\Desktop\dsa"
PS C:\Users\NISCHAL\OneDrive\Documents\Desktop\dsa> cd "c:\Users\NISCHAL\OneDrive\Documents\Desktop\dsa\" ; if ($?) { gcc doubleLinkList.c -o doubleLink
List } ; if ($?) { .\doubleLinkList }

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 1
Enter number of nodes: 2
Enter data for node 1: 20
Enter data for node 2: 30

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 2
Enter the value to insert left of: 10
Value not found

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 4
Doubly Linked List: 20 <-> 30 <-> NULL

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
```



```
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 4
Doubly Linked List: 20 <-> 30 <-> NULL

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 3
Enter value to delete: 20
Node deleted successfully

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 4
Doubly Linked List: 30 <-> NULL

--- Doubly Linked List Menu ---
1. Create List
2. Insert Node to Left
3. Delete Node by Value
4. Display List
5. Exit
Enter your choice: 
```