Lab program 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>


/* Structure of Doubly Linked List Node */

struct node

{

   int data;

   struct node *prev;

   struct node *next;

};


struct node *head = NULL;


/* Function Prototypes */

void create();

void insert_left();

void delete_value();

void display();


/* Main Function */

int main()

{

   int choice;


   while (1)

   {
```

```
    printf("\n\n----- DOUBLY LINKED LIST MENU -----");
    printf("\n1. Create Doubly Linked List");
    printf("\n2. Insert a node to the left of a node");
    printf("\n3. Delete a node based on value");
    printf("\n4. Display list");
    printf("\n5. Exit");
    printf("\nEnter 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:
       exit(0);
    default:
       printf("\nInvalid choice!");
    }
}
```

```
    return 0;
}

/* Create Doubly Linked List */
void create()
{
    struct node *newnode, *temp;
    int n, i;

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

    for (i = 0; i < n; i++)
    {
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Enter data: ");
        scanf("%d", &newnode->data);

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

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

```
            newnode->prev = temp;
            temp = newnode;
        }
    }
    printf("Doubly linked list created successfully.");
}

/* Insert a Node to the Left of a Given Node */
void insert_left()
{
    struct node *newnode, *temp;
    int key;

    if (head == NULL)
    {
        printf("\nList is empty.");
        return;
    }

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

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

    if (temp == NULL)
    {
        printf("\nNode not found.");
```

```
        return;
    }

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

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

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

    temp->prev = newnode;

    printf("Node inserted successfully.");
}

/* Delete Node Based on Specific Value */
void delete_value()
{
    struct node *temp;
    int key;

    if (head == NULL)
    {
        printf("\nList is empty.");
```

```
        return;
    }

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

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

    if (temp == NULL)
    {
        printf("\nValue not found.");
        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.");
}

/* Display Doubly Linked List */
```

```
void display()
{
   struct node *temp;

   if (head == NULL)
   {
      printf("\nList is empty.");
      return;
   }

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

```
----- DOUBLY LINKED LIST MENU -----
1. Create Doubly Linked List
2. Insert a node to the left of a node
3. Delete a node based on value
4. Display list
5. Exit
Enter your choice: 1

Enter number of nodes: 4
Enter data: 12
Enter data: 13
Enter data: 14
Enter data: 15
Doubly linked list created successfully.

----- DOUBLY LINKED LIST MENU -----
1. Create Doubly Linked List
2. Insert a node to the left of a node
3. Delete a node based on value
4. Display list
5. Exit
Enter your choice: 2

Enter value to insert to the left of: 12
Enter new data: 32
Node inserted successfully.

----- DOUBLY LINKED LIST MENU -----
1. Create Doubly Linked List
2. Insert a node to the left of a node
3. Delete a node based on value
4. Display list
5. Exit
Enter your choice: 4

Doubly Linked List: 32 <-> 12 <-> 13 <-> 14 <-> 15 <-> NULL
```

```
----- DOUBLY LINKED LIST MENU -----
1. Create Doubly Linked List
2. Insert a node to the left of a node
3. Delete a node based on value
4. Display list
5. Exit
Enter your choice: 3

Enter value to delete: 4

Value not found.

----- DOUBLY LINKED LIST MENU -----
1. Create Doubly Linked List
2. Insert a node to the left of a node
3. Delete a node based on value
4. Display list
5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 27.028 s
Press any key to continue.
```