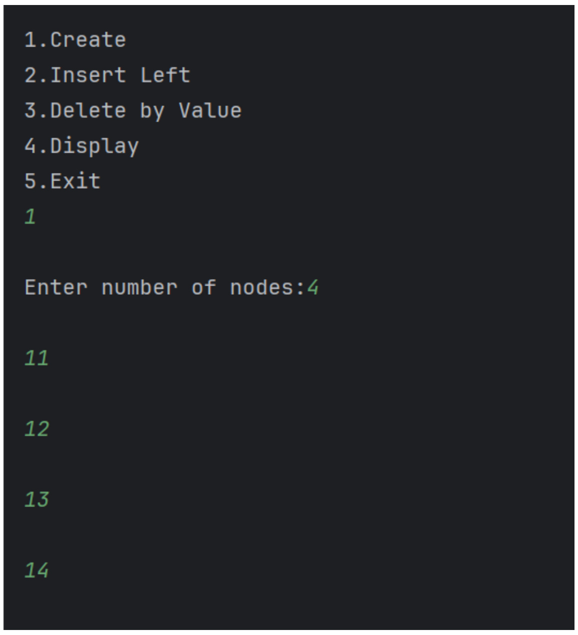
LAB PROGRAM 7

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

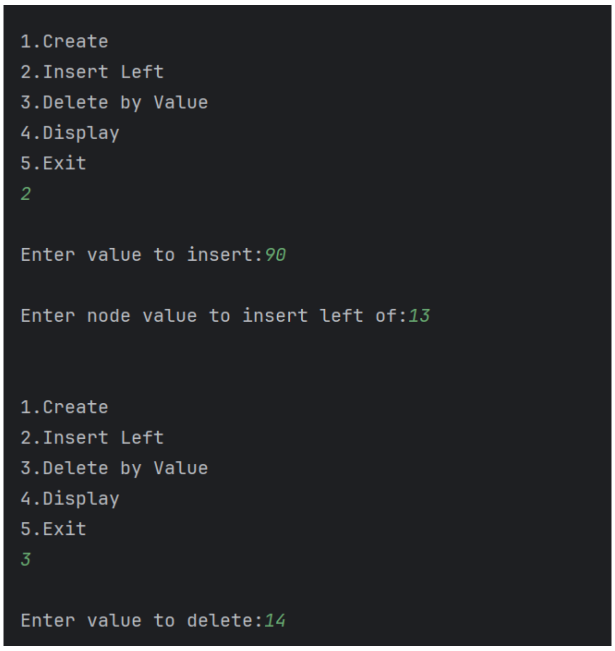
```
#include <stdio.h>

#include <stdlib.h>


struct node {

   int data;

   struct node *prev;

   struct node *next;

};


struct node *head = NULL;


void create() {

   int n, x;

   struct node *newnode, *temp;


   printf("Enter number of nodes: ");

   scanf("%d", &n);


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

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

      scanf("%d", &x);

      newnode->data = x;

      newnode->prev = NULL;
```

```
        newnode->next = NULL;

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

void insertLeft() {
    int key, x;
    struct node *temp, *newnode;

    if (head == NULL)
        return;

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

    temp = head;

    while (temp != NULL && temp->data != key)
        temp = temp->next;
    if (temp == NULL) {
        printf("Node not found\n");
        return;
    }

    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = x;

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

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

    temp->prev = newnode;
}

void deleteValue() {
    int key;
    struct node *temp;

    if (head == NULL)
        return;

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

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

    if (temp == NULL) {
        printf("Node 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);
}

void display() {
    struct node *temp = head;

    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice;

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

        switch (choice) {
            case 1: create(); break;
            case 2: insertLeft(); break;
            case 3: deleteValue(); break;
            case 4: display(); break;
            case 5: exit(0);
            default: printf("Invalid choice\n");
        }
    }
}
```

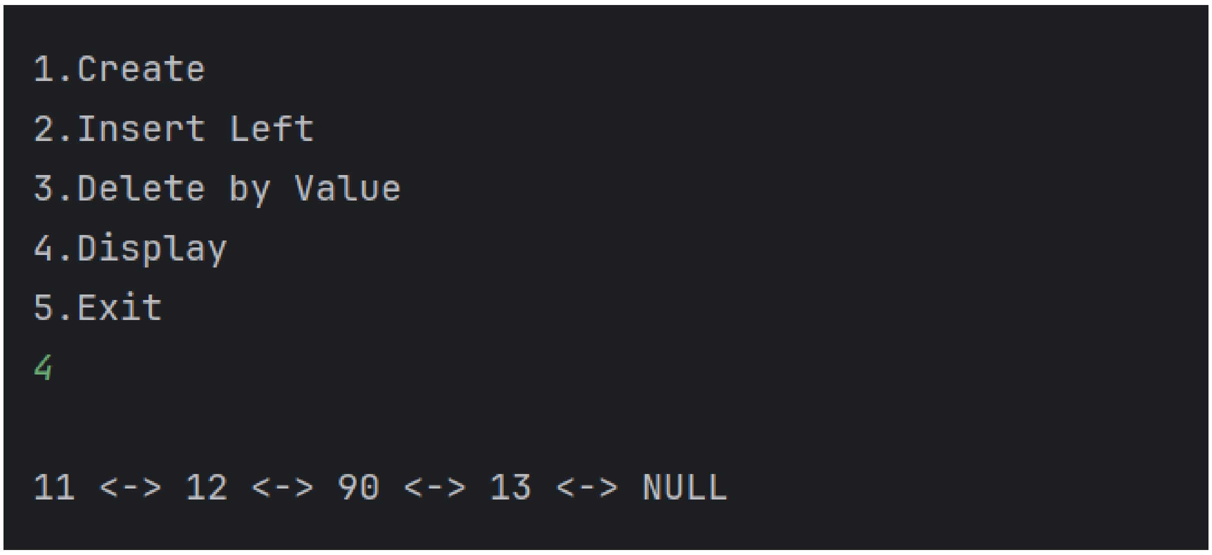
OUTPUT:



```
1.Create
2.Insert Left
3.Delete by Value
4.Display
5.Exit
1

Enter number of nodes:4

11

12

13

14
```



```
1.Create
2.Insert Left
3.Delete by Value
4.Display
5.Exit
2

Enter value to insert:90

Enter node value to insert left of:13


1.Create
2.Insert Left
3.Delete by Value
4.Display
5.Exit
3

Enter value to delete:14
```



```
1.Create
2.Insert Left
3.Delete by Value
4.Display
5.Exit
4


11 <-> 12 <-> 90 <-> 13 <-> NULL
```