Lab 11th 4a) WAP to Implement doubly ... 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 specific value.

d) Display the contents to the list:

Pseudocode:

```
struct Node
    int data
    struct Node * prev * next

function createlist
        int i, data
        struct Node * newNode
for(i= 1; i<=n; i+1)
    printf (Enter data)

newNode = (struct Node*) malloc (sizeof
                                 (struct Node));
    newNode -> data = data;
newNode -> prev = newNode -> next = NULL;
if head == NULL
    head = tail = newNode
else
    tail -> next = newNode
    newNode -> prev = tail
    tail = newNode
```

function Insert At End

```
Struct Node * newNode = (struct Node *) malloc
                        (sizeof (Struct Node));
newNode -> data = data.
newNode -> next = NULL
newNode -> prev = tail
if (tail == NULL)
    head = tail = newNode
else
    tail -> next = newNode
    tail = newNode
```

function delete by value

```
Struct Node * temp = head;
if (head == NULL)
  print (list Empty )
  return
while temp != NULL && temp -> data != value
    temp = temp -> next
if (temp == NULL)
    printf ("value not found")
    return
if (temp == head )
  delete At front()
  else if (temp = tail)
      delete At End ()
  else
  temp -> prev -> next = temp -> next
  temp -> prev -> prev = temp -> prev
```

```
free (temp)
function display (){
        Struct Node * temp = head;
        printf ("List (forward): ")
        while (temp! = NULL);
          print f (" %d <-> "; temp -> data)
            temp = temp ->next;
      print ("NULL");
```

MG.

node;

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

Struct Node {
      int data ;
       struct Node *prev , * next;
  };

Struct Node  * head = NULL;
Struct Node  * tail = NULL;

void createList (int n) {
     int i; data ;
     struct Node * newNode;

for (i = 1; i <= n; i++ ) {
       printf ("Enter data for node %d : ",i);
       scanf ("%d", & data);
```

```
newNode = (struct Node*) malloc (size of (struct Node));
newNode => data = data;
new Node -> prev = new Node -> next = NULL;
if (head == NULL) {
    head = tail = newNode;
} else {
    tail -> next = newNode;
    new Node -> prev = tail;
    tail = new Node;
}
}
void insertAtFront (int data) {
    Struct Node * newNode = (Struct Node*)
                malloc (size of (struct Node));
    new Node -> data = data;
void insertLeft (int Key, int data) {
    struct node * temp = head;
    Struct node * newNode;
    while (temp != NULL && temp -> data != value) {
        temp = temp -> next;
    }
    if (temp == NULL) {
        printf ("Value %d not found. \n", value);
        return;
    }
```

```
newNode = (struct Node *) malloc (sizeof (struct Node));
newNode -> data = data;

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

void deleteValue (int value) {
    struct Node *temp = head;

while (temp != NULL && temp -> data != value) {
    temp = temp -> next;
}

if (temp == head) {
    head = head -> next;
    if (head != NULL)
        head -> prev = NULL;
    free (temp);
    return ;
}
if (temp == tail) {
    tail = tail -> prev;
    tail -> next = NULL;
    free (temp);
    return ;
}
```

```
temp->prev->next = temp->next;
temp->next->prev = temp->prev;
free(temp);
}
void display() {
    struct Node *temp = head;
    printf("\nDoubly Linked List: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int n, choice, val, data;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    createList(n);
    display();
    while (1) {
        printf("\n1. InsertLeft\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                printf("Enter existing value: ");
                scanf("%d", &key);
                printf("Enter data: ");
                scanf("%d", &data);
                insertLeft(key, data);
                break;
```

```
case 2:
    printf("Enter value to delete:");
    scanf("%d", &Key);
    deleteValue(Key);
    break;

case 3:
    display();
    break;

case 4:
    exit(0);

  }
}
  return 0;

}
```

Output:

```
Enter number of nodes: 4
Enter data for node 1: 1
Enter data for node 2: 2
Enter data for node 3: 3
Enter data for node 4: 4

List: 1 2 3 4

1. Insert left.
2. Delete.
3. Display.
4. Exit.
Enter choice: 1
Enter existing value: 1
Enter data: 9

1. Insert Left
2. Delete
3. Display
4. Exit
Enter choice: 2
Enter value to delete 3

1. Insert Left
2. Delete
3. Display
4. Exit
Enter choice: 3
List: 9 1 2 4

1. Insert Left
2. Delete
3. Display
4. Exit
Enter choice: 4
Process returned (0).
```