4) CPP to Implement doubly linked 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.

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

struct node {
int data;
 struct node *prev, *next;
};

struct Node * CreateNode (int value)
{
struct Node * new_node = (struct Node *) malloc (size of
 (struct Node));
  new_node -> data = value;
  new_node -> prev = null;
  new_node -> next = null;
  return new_node;
}

void displaylist (struct node * head) {
  while (head != NULL)
  {
  printf ("%d ", & head -> data);
    head = head -> next;
}
  printf ("Null \n");
}
```

```
void insertLeft (struct node ** head, int value, int
    target value)
{
 struct Node * new_node = create Node (value);
 struct Node * current = *head;

while (current != NULL && current -> data != target
 value)
{
  current = current -> next;
}
if (current != NULL)
{
  new_node -> prev = current -> prev;
  new_node -> next = current;
  if (current -> prev != NULL)
  {
      current -> prev -> next = new_node;
  }
  Else
  {
    * head = new node;
  }
    current -> prev = new node;
}
Else {
    printf ("Target node with value %d not found
 \n", target value);
}
}
```

```
void deleteNode (struct Node ** head, int value)
{
struct Node * current = *head ;

while (current != NULL && current -> data != value)
{
  current = current -> next ;
}
 if (current != NULL)
 {
  if (current -> prev != NULL)
  {
   current -> prev -> next = current -> next ;
  }
  else {
  *head = current -> next ;
  }
  if (current -> next != NULL)
  {
   current -> next -> prev = current -> prev ;
  }
  free (current) ;
 }
 else
 {
  printf ("Node with value %d not found..\n", value);
 }
}
```

```
int main () {
struct Node * head = Create Node (1);
head -> next = Create Node(2);
head -> next -> prev = head;
head -> next -> next = Create Node (3);
head -> next -> next -> prev = head -> next;

printf(" original Doubly linked list : \n");
display list (head);

Insert Left (&head, 4, 2);
printf("\n list after inserting 4 to the left of 2: \n");
display list (head);

deleting (&head, 2);
printf("\n list after deleting node with value 2: \n");
display (head);

return (0);
}
```

Output:

original doubly linked list :
1 <--> 2 <--> 3 <--> Null

list after inserting 4 to the left of 2:
1 <--> 4 <--> 2 <--> 3 <--> Null;

list after deleting node with value 2 :
1 <--> 4 <--> 3 <--> NULL