9. Write a program to implement doubly link list with primitive operations a) a) Create a doubly linked list. b) Insert a new node to the left of the node. b) c) Delete the node based on a specific value. c) Display the contents of the list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *rlink;
   struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
   NODE x;
   x = (NODE)malloc(sizeof(struct node));
   if (x == NULL)
   {
      printf("Memory full\n");
      exit(0);
   }
   return x;
}
void freenode(NODE x)
```

```
{
   free(x);
}

NODE create(int item, NODE head)
{
   NODE temp, cur;
   temp = getnode();
   temp->info = item;
   temp->llink = NULL;
   temp->rlink = NULL;
   cur = head->llink;
   head->llink = temp;
   temp->rlink = head;
   cur->rlink = temp;
   temp->llink = cur;
   return head;
}

void ddisplay(NODE head)
{
   NODE temp;
   if (head->rlink == head)
   {
```

```
        printf("List is empty\n");
    }
    printf("The contents of the list are:\n");
    temp = head->rlink;
    while (temp != head)
    {
        printf("%d ", temp->info);
        temp = temp->rlink;
    }
}

NODE dinsert_leftpos(int item, NODE head)
{
    NODE cur, prev, temp;
    if (head->rlink == head)
    {
        printf("List is empty\n");
        return head;
    }
    cur = head->rlink;
    while (cur != head)
    {
        if (cur->info == item)
        {
```

```
            break;
        }
        cur = cur->rlink;
    }
    if (cur == head)
    {
        printf("No such item found in the list\n");
        return head;
    }
    prev = cur->llink;
    temp = getnode();
    temp->llink = NULL;
    temp->rlink = NULL;
    printf("Enter the item to be inserted at the left of the given
item:\n");
    scanf("%d", &temp->info);
    prev->rlink = temp;
    temp->llink = prev;
    temp->rlink = cur;
    cur->llink = temp;
    return head;
}

NODE delete_all_key(int item, NODE head)
```

```
{
   NODE prev, cur, next;
   int count;
   if (head->rlink == head)
   {
      printf("LE");
      return head;
   }
   count = 0;
   cur = head->rlink;
   while (cur != head)
   {
      if (item != cur->info)
         cur = cur->rlink;
      else
      {
         count++;
         prev = cur->llink;
         next = cur->rlink;
         prev->rlink = next;
         next->llink = prev;
         freenode(cur);
         cur = next;
      }
```

```
    }

    if (count == 0)
        printf("Key not found");
    else
        printf("Key found at %d positions and are deleted\n", count);

    return head;
}
int main()
{
    NODE head;
    int item, choice, key;
    head = getnode();
    head->llink = head;
    head->rlink = head;
    for (;;)
    {
        printf("\n1:\tCreate\n2:\tdisplay\n3:\tinsert
lestpos\n4:\tdelete_based on specified value\n5:\texit\n");
        printf("\nEnter the choice->\n");
        scanf("%d", &choice);
        switch (choice)
        {
```

```
case 1:
   printf("Enter the item :\n");
   scanf("%d", &item);
   head = create(item, head);
   break;

case 2:
   ddisplay(head);
   break;

case 3:
   printf("Enter the key element:\n");
   scanf("%d", &key);
   head = dinsert_leftpos(key, head);
   break;

case 4:
   printf("Enter the key value\n");
   scanf("%d", &item);
   delete_all_key(item, head);
   break;
case 5:
   exit(0);
default:
```

```
        printf("Invalid choice\n");
      }
   }
   return 0;
}
```

Output:

```
1:      Create
2:      display
3:      insert lestpos
4:      delete_based on specified value
5:      exit

Enter the choice->
1
Enter the item :
4

1:      Create
2:      display
3:      insert lestpos
4:      delete_based on specified value
5:      exit

Enter the choice->
3
Enter the key element:
4
Enter the item to be inserted at the left of the given item:
5

1:      Create
2:      display
3:      insert lestpos
4:      delete_based on specified value
5:      exit

Enter the choice->
2
The contents of the list are:
5 4
```

```
1:      Create
2:      display
3:      insert lestpos
4:      delete_based on specified value
5:      exit

Enter the choice->
4
Enter the key value
4
Key found at 1 positions and are deleted

1:      Create
2:      display
3:      insert lestpos
4:      delete_based on specified value
5:      exit

Enter the choice->
2
The contents of the list are:
5
1:      Create
2:      display
3:      insert lestpos
4:      delete_based on specified value
5:      exit

Enter the choice->
5
```