

DS Lab Program 9 -

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

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

```
struct Node {
```

```
    int info;
```

```
    struct Node *link, *llink;
```

```
};
```

```
typedef struct Node node;
```

```
node getnode () {
```

```
    node x;
```

```
    x = (node) malloc (sizeof (Node));
```

```
    if (x == NULL) {
```

```
        printf ("Memory full\n");
```

```
        exit (0);
```

```
}
```

```
return x;
```

```
}
```

```
void freenode (node n) {
```

```
    free (n);
```

```
}
```

```
node insert_front (int item, node head) {
```

```
    node temp, cur;
```

```
    temp = getnode ();
```

```
    temp -> info = item;
```

```
    temp -> link = NULL;
```

```
    cur = head -> link;
```

```
head -> link = temp
```

```
head -> link = temp;
```

```
temp -> link = cur;
```

```
cur -> link = temp;
```

```
return head;
```

```
}
```

Date / /

```
node insert_rear (int item, node head) {
    node temp = getnode(), cur;
    temp->info = item;
    cur = head->link;
    head->link = temp;
    temp->link = cur;
    cur->link = temp;
    return head;
}
```

```
node delete_front (node head) {
    node cur, next;
    if (head->link == head) {
        printf ("Empty\n");
        return head;
    }
    cur = head->link;
    next = cur->link;
    head->link = next;
    next->link = head;
    printf ("Item deleted = %d\n", cur->info);
    free (node (cur));
    return head;
}
```

```
node delete_rear (node head) {
    node cur, prev;
    if (head->link == head) {
        printf ("Empty\n");
        return head;
    }
    cur = head->link;
    prev = cur->link;
    head->link = prev;
    free (node (cur));
    return head;
}
```

```
prev->rlink = head;
printf ("Item deleted = %d\n", cur->info);
freemode (cur);
return head;
```

y

```
node insert_leftpos (int item, node head) {
    node temp, cur, prev;
    if (head->rlink == head) {
        printf ("Empty\n");
        return head;
    }
```

y

```
cur = head->rlink;
while (cur != head) {
    if (item == cur->info)
        break;
    cur = cur->rlink;
}
```

y

```
if (cur == head)
    printf ("Key not found\n");
return head.
```

y

```
prev = cur->rlink;
printf ("Enter towards left of %d", item);
temp = getnode();
scanf ("%d", temp->info);
prev->rlink = temp;
temp->llink = prev;
cur->llink = temp;
temp->rlink = cur;
return head;
```

y

```
node delete_value (int item, node head) {
```

```
node prev, cur, next;
```

```
int count;
```

```
if (head->slink == head) {  
    printf ("List is empty \n");  
    return head;
```

```
}  
count = 0;
```

```
cur = head->slink;
```

```
while (cur != head) {
```

```
    if (item != cur->info)
```

```
        cur = cur->slink;
```

```
    else {
```

```
        count++;
```

```
        prev = cur->llink;
```

```
        next = cur->slink;
```

```
        prev->slink = next;
```

```
        next->llink = prev;
```

```
        freeNode (cur);
```

```
}
```

```
if (count == 0)
```

```
    printf ("Key not found \n");
```

```
else {
```

```
    printf ("Key found at %d. position and it  
    deleted \n", count);
```

```
    return head;
```

```
}  
void display (node head) {
```

```
node temp;
```

```
if (head->slink == head) {
```

```
    printf ("Empty \n");
```

```
return;
```

```
}
```

```
printf ("Contents of Queue \n");
temp = head->link;
while (temp != head) {
    printf ("%d\n", temp->info);
    temp = temp->link;
```

```
void main () {
```

```
node head, last;
```

```
int item, choice;
```

```
head = getnode();
```

```
head->link = head;
```

```
head->link = head;
```

```
for (;;) {
```

```
    printf ("1. Insert front \n 2. Insert rear \n 3
```

```
    Delete front \n 4. Delete rear \n 5. Insert left
```

```
position \n 6. Delete specified value \n 7. Display \n");
```

```
scanf ("%d", &choice);
```

```
switch (choice) {
```

```
case 1: printf ("Enter item \n");
```

```
scanf ("%d", &item);
```

```
last = insert_front (item, head);
```

```
break;
```

```
case 2: printf ("Enter item \n");
```

```
scanf ("%d", &item);
```

```
last = insert_rear (item, head);
```

```
break;
```

```
case 3: last = delete_front (head);
```

```
break;
```

```
case 4: last = delete_rear (head);
```

```
break;
```

```
case 5: printf ("Enter key item \n");
```

```
scanf ("%d", &item);
```

```
head = insert_left_pos(item, head);
break;
case 6: printf ("Enter key item\n");
scanf ("%d", & item);
head = delete_value(item, head);
break;
case 7: display (head);
break;
default: exit (0);
3'3'5,
```

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 struct node{
4     struct node *rlink;
5     int info;
6     struct node *llink;
7 };
8 typedef struct node *NODE;
9 NODE getnode(){
10     NODE x=(NODE)malloc(sizeof(struct node));
11     if(x==NULL){
12         printf("Cannot allocate Memory\n");
13         exit(0);
14     }
15     return x;
16 }
17 NODE insert_reard(NODE head,int item){
18     NODE temp=getnode();
19     temp->info=item;
20     temp->rlink=NULL;
21     temp->llink=NULL;
22     NODE cur=head->llink;
23     head->llink=temp;
24     temp->rlink=head;
25     temp->llink=cur;
26     cur->rlink=temp;
27     return head;
28 }
29 NODE insert_left(NODE head,int key){
30     if(head==head->rlink){
31         printf("List is empty\n");
32         return head;
33     }
34     NODE cur=head->rlink;
35     while(cur!=head){
36         if(cur->info==key)
37             break;
38         cur=cur->rlink;
39     }
40     if(cur==head){
41         printf("Element not found\n");
42         return head;
43     }
44     NODE prev=cur->llink;
45     NODE temp=getnode();
```

```
44     NODE prev=cur->llink;
45     NODE temp=getnode();
46     printf("Enter the item\n");
47     scanf("%d",&temp->info);
48     temp->rlink=cur;
49     temp->llink=prev;
50     prev->rlink=temp;
51     cur->llink=temp;
52     return head;
53 }
54 NODE delete_infod(NODE head,int item){
55     if(head==head->rlink){
56         printf("List is empty\n");
57         return head;
58     }
59     NODE cur=head->rlink;
60     while(cur!=head){
61         if(cur->info==item)
62             break;
63         cur=cur->rlink;
64     }
65     if(cur==head){
66         printf("Element not found in the list\n");
67         return head;
68     }
69     NODE prev=cur->llink;
70     NODE next=cur->rlink;
71     prev->rlink=next;
72     next->llink=prev;
73     printf("Deleted item = %d\n",cur->info);
74     free(cur);
75     return head;
76 }
77 void displayd(NODE head){
78     if(head==head->rlink){
79         printf("List is empty\n");
80         return;
81     }
82     NODE cur=head->rlink;
83     while(cur!=head){
84         printf("%d\n",cur->info);
85         cur=cur->rlink;
86     }
87 }
88 int main(){
```

```
88 int main(){
89     int item,ch;
90     NODE head=getnode();
91     head->rlink=head;
92     head->llink=head;
93     while(1){
94         printf(" 1:Insert item\n 2:Insert at left position\n 3:Delete it");
95         printf("Enter the choice\n");
96         scanf("%d",&ch);
97         switch(ch){
98             case 1:printf("Enter the item\n");
99                 scanf("%d",&item);
100                head=insert_reard(head,item);
101                break;
102            case 2:printf("Enter the key to whose left item should be inserted\n");
103                scanf("%d",&item);
104                head=insert_left(head,item);
105                break;
106            case 3:printf("Enter the element to be deleted\n");
107                scanf("%d",&item);
108                head=delete_infod(head,item);
109                break;
110            case 4:displayd(head);
111                break;
112            default:exit(0);
113        }
114    }
115 }
116
117 }
```

```
Enter the item
50
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
4
10
20
40
50
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
2
Enter the key to whose left item should be inserted
40
Enter the item
30
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
4
10
20
30
40
50
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
```

2
Enter the key to whose left item should be inserted
40
Enter the item
60
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
3
Enter the element to be deleted
60
Deleted item = 60
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
3
Enter the element to be deleted
40
Deleted item = 40
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
4
10
20
30
50
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice

```
5:Exit
Enter the choice
4
10
20
30
50
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
3
Enter the element to be deleted
50
Deleted item = 50
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
3
Enter the element to be deleted
20
Deleted item = 20
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
4
10
30
1:Insert item
2:Insert at left position
3:Delete item
4:Display
5:Exit
Enter the choice
```