Name: Shreechasi Kulkarni
USN: 1BM19CS153

```
#include<stdio.h>
#include<math.h>
#include<stdio.h>
struct node
{
    int info
    struct node *clink;
    struct node *rlink;
};
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 insert_front(int item, NODE head)
{
```

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

NODE delete_front(NODE head)
{
    NODE cur, next;
    if (head -> rlink == head)
    {
        printf("Empty list\n");
        return head;
    }
```

```
cur = head -> rlink;
next = cur -> rlink;
head -> rlink = -> next;
next -> llink = -> head;
printf("The node deleted is %d\n", cur
free(cur);
return head
}

NODE delete_rear(NODE head)
{
    NODE cur, prev;
    if(head -> rlink == head)
    {
        printf("Empty");
        return head;
    }
    cur = head -> llink;
    prev = cur -> llink;
    head -> llink = prev;
    prev -> rlink = head;
    printf("Node deleted is %d\n", cur->info)
    free(cur);
    return head;
}
void display(NODE head)
{
    NODE temp;
    if(head -> rlink == head)
    {
        printf("Empty\n");
        return;
    }
```

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

NODE insert_leftpos(int item, NODE head)
{
    NODE temp, cur, prev;
    if (head -> rlink == head)
    {
        printf ("Empty \n");
        return head;
    }
    cur = head -> rlink;
    while (cur != head)
    {
        if (item == cur -> info)
            break;
        cur = cur -> rlink;
    }
    if (cur == head)
        printf ("Key not found \n");
    return head;
}
```

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

NODE insert_right_pos(int item, NODE head

15 x 16 + 15 x 16

240 + 15

```
{
  NODE temp, cur, prev
  if (head -> rlink == head)
  {
      printf("empty ln");
      return head
  }
  cur = head -> rlink;
  while (cur != head)
  {
      if (item == cur -> info)
          break;
      cur = cur -> rlink.
  }
  if (cur == head)
      printf("key not found ln");
      return head

}
```

```
prev = cur -> llink
printf("Enter toward right of %d = ", item)
temp = getnode();
Scanf("%d", & temp -> info);
temp -> rlink = cur -> rlink
temp -> llink = cur
cur -> rlink = temp
return head
}

Void search(NODE head)
{
    NODE ptr
    int item, i=0, flag;
    ptr = head;
    if (ptr == NULL)
    {
        printf("Empty list\n");
    }
    else
    {
        printf("Enter item to search\n");
        Scanf("%d", & item);
        while (ptr != NULL)
        {
            if (ptr -> info == item)
            {
                printf("Item found at %d\n", i);
                flag = 0;
                break
            }
            else
```

```
        {
            flag = 1
        {
        i++;
        ptr = ptr -> rlink
    }
    if (flag == 1)
    {
        printf("\nItem not found\n");
        break;
}
```

```
NODE delete_all(int item, NODE head)
{
    NODE prev, cur, next;
    int count;
    if(head -> rlink == head)
    {
        printf("Empty\n");
        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;
            free(cur);
            cur = next;
        }
    }
    if(count == 0)
        printf("Not found\n");
    else
        printf("Key found at %d position\n", count);
    return head;
}
```

```
int main()
{
    NODE head, last
    int item, choice
    head = getnode()
    head->rlink = head
    head->llink = head
    for(;;)
    {
        printf("1: Insert front\n");
        printf("2: Insert Rear\n");
        printf("3: Delete front\n");
        printf("4: DeleteRear\n");
        printf("5: display\n");
        printf("6: insert left\n");
        printf("7: Remove Duplicates\n");
        printf("8: search\n");
        printf("9: insertRight\n");
        printf("10: exit\n");
        printf("Enter your choice\n");
        scanf("%d", &choice)
        switch(choice)
        {
            case 1: printf("Enter item\n");
                    scanf("%d", &item);
                    last = dinsert_front(item, head);
                    break

            case 2: printf("Enter item\n");
                    scanf("%d", &item);
                    last = insert_end(item, head);
                    break;
```

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

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

case 5: display (head)
        break;

case 6: printf ("Enter the item \n");
        scanf ("%d", &item);
        head = insert_leftpos (item, head);
        break;

case 7: printf ("Enter the item \n");
        scanf ("%d", &item);
        head = delete_all_key (item, head);
        break;

case 8: search (head);
        break;

case 9: printf ("Enter item \n");
        scanf ("%d", &item);
        head = insert_rightpos (item, head);
        break;

    default: exit (0);

}
}
```