Lab 9

```
#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;
}
NODE dinsert_rear (int item, NODE head)
{
    NODE temp, cur;
    temp = getnode();
    temp -> info = item;
```

```
        cur = head -> llink;
        temp -> llink = cur;
        cur -> rlink = temp;
        head -> llink = temp;
        temp -> rlink = temp;
        return head;
}
NODE dinsert_front (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 ddelete_front (NODE head)
{
    NODE cur, next;
    if (head -> rlink == head)
    {
        printf ("dqe empty \n");
        return head;
    }
```

```
        cur = head -> llink;
        prev = cur -> llink;
        head -> llink = cur;
        prev -> llink = head;
        printf ("the item deleted is %d \n", cur -> info);
        free (cur);
        return head;
    }
NODE ddelete_rear (NODE head)
    {
        NODE cur, prev;
        if (head -> rlink == head)
        {
            printf ("dq empty \n");
            return head;
        }
        cur = head -> llink;
        prev = cur -> llink;
        head -> llink = prev;
        prev -> rlink = head;
        printf ("the item deleted is %d \n", cur->info);
        free (cur);
        return head;
}
```

```
NODE lsearch (NODE head, int key, int z)
{
    NODE cur, prev, temp;
    int f=0, c=1;
    if (head->rlink == head)
    {
        printf("list empty\n");
        return head;
    }
    cur = head->rlink;
    while (cur != head)
    {
        if (cur->info == key)
        {
            f=1;
            break;
        }
        cur = cur->rlink;
        c++;
    }
    if (f==1 && z==0)
    {
        printf("search successful, found at index :%d\n",c
        return head;
    }
```

```
if (f==1 && z==1)
{
    prev = cur -> llink;
    printf ("enter towards left of %d = ", key);
    temp = getnode();
    scanf ("%d", &temp -> info);
    prev -> rlink = temp;
    temp -> llink = temp;
    temp -> rlink = cur;
    return head;
}
if (f==1 && z==2)
{
    prev = cur;
    cur = cur -> rlink;
    printf ("enter towards right of %d:", key)
    temp = getnode();
    scanf ("%d", &temp -> info);
    prev -> rlink = temp;
    temp -> llink = prev;
    cur -> llink = temp;
    return head;
}
printf ("search unsuccessful\n");
}
```

```
NODE delete_all_key (int item, NODE head)
{
    NODE prev, cur, next;
    int count;
    if (head->rlink == head)
    {
        printf ("List 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 ("key not found \n");
else
    printf ("key found at %d position and are deleted \n",
            count);
return head;
}
void display (NODE head)
{
    NODE temp;
    if (head -> rlink == head)
    {
        printf ("dq empty \n");
        return;
    }
    printf ("contents of dq \n");
    temp = head -> rlink;
    while (temp != head)
    {
        printf ("%d", temp -> info);
        temp = temp -> rlink;
    }
    printf ("\n");
}
```

```
void main()
{
  NODE head, last;
  int item, choice;
  head = getnode();
  head -> rlink = head;
  head -> llink = head;
  for(;;)
  {
    printf("Enter choice: \n1. Insert Front \n2. Delete
        Front \n3. Insert Rear \n4. Delete Rear \n5.
        Simple search \n6. Insert left of key \n7.
        Insert right of key \n8. Delete all occurance \n9.
        Display \n - Any other key to exit-\n");
    scanf("%d", &choice);
    switch(choice)
    {
      case 1: printf("Enter the item at front end \n");
              scanf("%d", &item);
              head = dinsert_front(item, head);
              break;
      case 2: head = ddelete_front(head);
      case    break;
      case 3: printf("enter the items at rear
                end \n");
```

```
        scanf("%d", &item);
        head = dinsert_rear(item, head);
        break;
    case 4: head = ddelete_rear(head);
        break;
    case 5: printf("Enter key \n");
        scanf("%d", &item);
        head = dsearch(head, item, 0);
        break;
    case 6: printf("Enter key \n");
        scanf("%d", &item);
        head = dsearch(head, item, 1); break;
    case 7: printf("Enter key \n");
        scanf("%d", &item);
        head = dsearch(head, item, 2);
        break;
    case 8: printf("Enter key \n");
        scanf("%d", &item);
        head = delete_all_key(item, head);
        break;
    case 9: display(head);
        break;
    default: exit(0);
    }
}
```