Lab - 9 - DLL

```
# 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;
```

Scanned by CamScanner

```
temp -> llink = cur;
cur -> rlink = temp;
head -> llink = temp;
temp -> rlink = head;
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("dq empty");
return head;
}
cur = head -> rlink;
next = cur -> rlink;
```

Scanned by CamScanner

```
head -> rlink = next;
next -> llink = head;
printf ("the item deleted is %-d\n", cur->info);
  free (cur);
  return head;
}
NODE delete_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;
    }
```

Scanned by CamScanner

```
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 = prev;
    cur -> llink = temp;
    temp -> rlink = cur;
    return head;
}
if (f==1 && z==2)
{
```

ANKIT IND / 205 X 260 / SINGLE LINE / 53 X 42 / 2 COL / 13-11-19

Scanned by CamScanner

```
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;
temp -> rlink = cur;
return head;
}
printf("search unsuccesful \n");
}
NODE delete_all_key (int item, NODE head)
{
NODE prev, cur, next;
int count;
if (head -> rlink == head)
{
printf("first empty \n");
return head;
}
count = 0;
cur = head -> rlink;
while (cur != head)
{
if (item != cur -> info)
cur = cur -> rlink;
```

Scanned by CamScanner

```
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 positions 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;
```

Scanned by CamScanner

```
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: \n 1. Insert front \n
                          2. Delete front \n
                          3. Insert rear \n
                          4. Delete rear \n
                          5. Simple Searching
                          6. Insert left of
                             key \n
                          7. Insert right \n
                          8. Insert left \n
                          9. Display");
case 1: printf("Enter the item at front end\n");
        scanf("%d", &item);
        head = dinsert-front(item, head);
        break;
```

ANKIT IND / 205 X 260 / SINGLE LINE / 53 X 42 / 2 COL / 12.11.19

Scanned by CamScanner

```
case 2:
        head = ddelete_front(head);
        break;
case 3:
        printf("enter the item at rear end\n");
        scanf("%d", &item);
        head = dinsert_rear(item, head);
        break;
case 4: 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 = dest_all_key(item, head);
        break;
case 9: display(head);
        break;
default: exit(0);
}
}
```

Scanned by CamScanner