Double-linked list

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

```
NODE dinsert_rear(int item, NODE head)
{
    NODE temp, cur;
    temp = getnode();
    temp -> info = item;
    cur = head -> llink;
    head -> llink = temp;
    temp -> llink = cur;
    temp -> rlink = head;
    cur -> rlink = temp;
    return head;
}
NODE dldelete_front(NODE head)
    NODE cur, next;
    if( head -> rlink == head)
    printf("dq empty\n");
    return head);
    cur = head -> rlink;
    next = cur -> rlink;
    head -> rlink = next;
    next -> llink = head;
    printf("item deleted %d", cur -> info);
    freenode(cur);
    return head;
}
```

```
NODE insert_leftpos(int item, NODE head)
{
 NODE temp, cur, prev;
 if (head->rlink == head)
 {
  printf("list 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");
  return head;
 }
 prev = cur->llink;
 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;
}
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;
{
  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
            & deleted \n", count);}
    return head;
}

Void search-info(int item, NODE head)
{
    NODE cur;
    if (head->rlink==head)
{
    printf("list empty \n");
    }
    cur = head->rlink;
while (cur!=head)
if (item==cur->info)
{
    printf("Search Successfully \n");
    break;
}
```

```
    cur = cur -> rlink;
  }
  if (cur == head)
  {
    printf ("Info not found \n");
  }
}
void display (NODE head)
{
  NODE temp;
  if (head -> rlink == head)
  printf("list empty \n");
    return;
}
for (temp = head -> rlink; temp != head;
     temp = temp -> rlink)
{
  printf ("%d \n", temp -> info);
}
Void main()
{
  int item, choice, key;
  NODE lasthead;
  head = getnode();
  head -> rlink = head;
```

```
for(;;)
{
printf("\n1. insert_rear\n 2. delete_front \n
        3. insert_key_pos left \n 4. delete all
        key \n 5. searching. display\n 7: exit");
printf( "Enter the choice\n");
Scanf("%d, & choice);
Switch (choice)
{
  case 1: printf ("Enter item\n");
          scanf("%d", & item);
          head = insert_ rear (head, item);
          break;
  case 2: last = dldelete_front (head);
          break;
  case 3: printf ("Enter the key\n");
          Scanf("%d", & item);
          head = delete
          head = insert_leftpos (item, head);
          break;
  Case 4; printf ("Enter the item\n");
          scanf("%d", & item);
          head = delete_all : key
                 (item, head);
```

```
case 5: printf("Enter key\n");
        scanf("%d", &item);
        search_info(item, head);
        break;
case 6: display(head);
        break;
default; exit(0);
    }
}
}
```