```
#include<stdio.h>
#include<conio.h>
#include<stdlib.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_front(int item, NODE hea
{
  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 dinsert_rear(int item, NODE head)
{
NODE temp, cur;
temp = getnode();
temp -> info = item;
cur = head -> llink;
head -> llink = temp;
temp -> rlink = head;
temp -> llink = cur;
cur -> rlink = temp;
return head;
}

NODE ddelete_front(NODE head)
{
NODE cur, next;
if (head -> rlink == head)
{
printf("double LL empty \n");
return head;
}
cur = head -> rlink;
next = cur -> rlink;
head -> rlink = next;
next -> llink = head;
printf("the node deleted is %d", cur->info);
freenode(cur);
return head;
}
```

```
NODE ddelete_rear(NODE head)
{
 NODE cur, prev;
if (head -> rlink == head)
{
   printf (" doubly LL empty \n");
return head;
}
cur = head -> dlink;
prev = cur -> dlink;
head -> dlink = prev;
prev -> rlink = head;
printf (" the node deleted is %d", cur -> in
freenode(cur);
 return head;
}
NODE dinsert_leftpos(int item, NODE he
{
 NODE temp, cur, prev;
if ( head -> rlink == head)
 {
   printf (" dist 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 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 insert_rightpos(int item, NODE head)
{
NODE temp, cur, next;
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\n");
return head;
}
```

```
next = cur -> rlink;
printf("enter towards right of %d=", item);
temp = getnode();
scanf("%d", &temp -> info);
cur -> rlink = temp;
temp -> llink = cur;
next -> llink = temp;
temp -> rlink = next;
return head;
}
NODE search(NODE head, int item)
{
NODE temp, cur;
int flag=0;
if (head -> rlink == head)
{
printf("list empty\n");
return head;
}
cur = head -> rlink;
while (cur != head)
{
if (item == cur -> info)
{
flag = 1;
break;
}
cur = cur -> rlink;
}
if (cur == head)
printf("search unsuccessful\n");
if (flag == 1)
printf("search successful\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 → link;
  {else
    count ++;
    prev = cur → llink;
    next = cur → rlink;
    prev → rlink = next;
     next → llink = prev;
    freenode (cur);
     cur = next;
  }
}
  if (count == 0)
    printf (" not Found \n");
  else
  printf (" Found at %d Positions & all deleted ", count);
  return head; }
```

```
void display(NODE head)
{
 NODE temp;
if (head->rlink == head)
{
printf("doubly LL empty \n");
return;
}
printf("contents of DLL \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("\n1.insertfront\n2:insertrear
\n3:deletefront\n4:deleterear\n5:insert
left of key element\n6:insert right of
key element\n7:simple search\n8:delete
repeating occurances\n9:display\n10:exit\n");
printf("enter the choice\n");
scanf("%d", &choice);
```

```
switch (choice)
{
case 1: printf("enter the item at the
        front end \n");
        scanf("%d", &item);
        last = dinsert_front(item, head);
        break;
Case 2: printf("enter the item at rear
                end|n");
        scanf("%d", &item);
        last = dinsert_rear(item, head);
        break;
Case 3: last = ddelete_front(head);
        break;
Case 4: last = ddelete_rear(head);
        break;
Case 5: printf("enter the key element|n
        scanf("%d", &item);
        last = insert_leftpos(item, head);
        break;
Case 6:
        printf("enter the key element \n");
        scanf("%d", &item);
        last = insert_rightpos(item, head);
        break;
Case 7:
        printf("enter the search element
        scanf("%d", &item);
        search(head, item);
        break;
Case 8: printf("enter element to
        be deleted|n");
        scanf("%d", &item);
```

```
last = delete_all_key(item, head);
break;
case 9: display(head);
break;
default: exit(0);
}
}
getch();
}
```