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_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;
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;
temp->llink=NULL;
temp->rlink=NULL;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
cur->rlink=temp;
temp->llink=cur;
return head;
}
NODE ddelete_front(NODE head)
{
NODE cur,next;
if (head->rlink==head)
{
printf("List is empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
printf("Item deleted at the front end is:%d\n",cur->info);
free(cur);
return head;
}
NODE ddelete_rear(NODE head)
{
NODE cur,prev;
if (head->rlink==head)
{
printf("List is empty\n");
return head;
}
cur=head->llink;
prev=cur->llink;
prev->rlink=head;
```

```
head->llink=prev;
printf("Item deleted at the rear end is:%d\n",cur->info);
free(cur);
return head;
}
void ddisplay(NODE head)
{
NODE temp;
if (head->rlink==head)
{
printf("List is empty\n");
}
printf("The contents of the list are:\n");
temp=head->rlink;
while (temp!=head)
{
printf("%d\n",temp->info);
temp=temp->rlink;
}
}
void dsearch(int key,NODE head)
{
NODE cur;
int count;
if (head->rlink==head)
{
printf("List is empty\n");
}
cur=head->rlink;
count=1;
while (cur!=head && cur->info!=key)
{
cur=cur->rlink;
count++;
}
if (cur==head)
{
printf("Search unsuccessfull\n");
}
else
```

```
{
printf("Key element found at the position %d\n",count);
}
}
NODE dinsert_leftpos(int item,NODE head)
{
NODE cur,prev,temp;
if (head->rlink==head)
{
printf("List is empty\n");
return head;
}
cur=head->rlink;
while (cur!=head)
{
if (cur->info==item)
{
break;
}
cur=cur->rlink;
}
if (cur==head)
{
printf("No such item found in the list\n");
return head;
}
prev=cur->llink;
temp=getnode();
temp->llink=NULL;
temp->rlink=NULL;
printf("Enter the item to be inserted at the left of the given item:\n");
scanf("%d",&temp->info);
prev->rlink=temp;
temp->llink=prev;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE dinsert_rightpos(int item,NODE head)
{
```

```
NODE temp,cur,next;
if (head->rlink==head)
{
printf("List is empty\n");
return head;
}
cur=head->rlink;
while (cur!=head)
{
if (cur->info==item)
{
break;
}
cur=cur->rlink;
}
if (cur==head)
{
printf("No such item found in the list\n");
return head;
}
next=cur->rlink;
temp=getnode();
temp->llink=NULL;
temp->rlink=NULL;
printf("Enter the item to be inserted at the right of the given item:\n");
scanf("%d",&temp->info);
cur->rlink=temp;
temp->llink=cur;
next->llink=temp;
temp->rlink=next;
return head;
}
NODE ddelete_duplicates(int item,NODE head)
{
NODE prev,cur,next;
int count=0;
if (head->rlink==head)
{
printf("List is empty\n");
return head;
```

```
}
cur=head->rlink;
while (cur!=head)
{
if (cur->info!=item)
{
cur=cur->rlink;
}
else
{
count++;
if (count==1)
{
cur=cur->rlink;
continue;
}
else
{
prev=cur->llink;
next=cur->rlink;
prev->rlink=next;
next->llink=prev;
free(cur);
cur=next;
}
}
}
if (count==0)
{
printf("No such item found in the list\n");
}
else
{
printf("Removed all the duplicate elements of the given item successfully\n");
}
return head;
}
int main()
{
NODE head;
```

```
int item, choice,key;
head=getnode();
head->llink=head;
head->rlink=head;
for(;;)
{
printf("\n1:dinsert front\n2:dinsert rear\n3:ddelete front\n4:ddelete
rear\n5:ddisplay\n6:dsearch\n7:dinsert lestpos\n8:dinsert rightpos\n9:ddelete
duplicates\n10:exit\n");
printf("enter the choice\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: printf("Enter the item at rear end:\n");
scanf("%d",&item);
head=dinsert_rear(item,head);
break;
case 3:head=ddelete_front(head);
  break;
case 4:head=ddelete_rear(head);
  break;
case 5:ddisplay(head);
  break;
   case 6:printf("Enter the key element to be searched:\n");
  scanf("%d",&key);
  dsearch(key,head);
  break;
   case 7:printf("Enter the key element:\n");
  scanf("%d",&key);
  head=dinsert_leftpos(key,head);
  break;
case 8:printf("Enter the key element:\n");
  scanf("%d",&key);
  head=dinsert_rightpos(key,head);
  break;
case 9:printf("Enter the key element whose duplicates should be removed:\n");
```

```
  scanf("%d",&key);
  head=ddelete_duplicates(key,head);
  break;
default:exit(0);
}
}
return 0;
}
```