main.c

```c
#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 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 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("dq empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
```

```
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("dq empty\n");
return head;
}
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is %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\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;
```

```
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;
```

```
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)
{
    flag=1;
    break;
}
cur=cur->rlink;
}
 if(cur==head)
 printf("search unsuccessfull\n");
 if(flag==1)
 printf("search successfull\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
```

```
        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 and are deleted",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)
{
```

```
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;
//clrscr();
for(;;)
{
    printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert left of k
    printf("enter the choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("enter the item at 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);
```

```
240         case 3:last=ddelete_front(head);
241             break;
242         case 4: last=ddelete_rear(head);
243             break;
244 
245         case 5:
246              printf("enter the key element\n");
247              scanf("%d",&item);
248              last=insert_leftpos(item,head);
249              break;
250          case 6:
251              printf("enter the key element\n");
252              scanf("%d",&item);
253              last=insert_rightpos(item,head);
254              break;
255         case 7:
256                 printf("enter the search element\n");
257                 scanf("%d",&item);
258                 search(head,item);
259                 break;
260         case 8: printf("enter element to be deleted\n");
261                 scanf("%d",&item);
262                 last=delete_all_key(item,head);
263         case 9: display(head);
264             break;
265         default:exit(0);
266         }
267     }
268 getch();
269 }
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
1
enter the item at front end
1

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
1
```

```
enter the choice
1
enter the item at front end
2

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
2
enter the item at rear end
3

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
```

```
8:delete repeating occurances
9:display
10:exit
enter the choice
2
enter the item at rear end
3

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
1
enter the item at front end
4

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
```

```
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
9
contents of dq
5421

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
7
enter the search element
4
search successfull
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
4
the node deleted is 1
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
9
```

```
10:exit
enter the choice
9
contents of dq
542

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
6
enter the key element
4
enter towards right of 4=3

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
```

```
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
9
contents of dq
5432

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
10


...Program finished with exit code 0
Press ENTER to exit console.
```