```c
#include<stdio.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;
```

main.c

```
35    return head;
36    }
37    NODE dinsert_rear(int item,NODE head)
38    {
39    NODE temp,cur;
40    temp=getnode();
41    temp->info=item;
42    cur=head->llink;
43    head->llink=temp;
44    temp->rlink=head;
45    temp->llink=cur;
46    cur->rlink=temp;
47    return head;
48    }
49    NODE ddelete_front(NODE head)
50    {
51    NODE cur,next;
52    if(head->rlink==head)
53    {
54    printf("dq empty\n");
55    return head;
56    }
57    cur=head->rlink;
58    next=cur->rlink;
59    head->rlink=next;
60    next->llink=head;
61    printf("the node deleted is %d",cur->info);
62    freenode(cur);
63    return head;
```

main.c

```
65  NODE ddelete_rear(NODE head)
66  {
67  NODE cur,prev;
68  if(head->rlink==head)
69  {
70  printf("dq empty\n");
71  return head;
72  }
73  cur=head->llink;
74  prev=cur->llink;
75  head->llink=prev;
76  prev->rlink=head;
77  printf("the node deleted is %d",cur->info);
78  freenode(cur);
79  return head;
80  }
81
82  NODE insert_leftpos(int item,NODE head)
83  {
84  NODE temp,cur,prev;
85  if(head->rlink==head)
86  {
87  printf("list empty\n");
88  return head;
89  }
90  cur=head->rlink;
91  while(cur!=head)
92  {
93  if(item==cur->info)break;
```

/ ShoddyPleasingSdk

Stop ■

main.c

```
122    {
123    if(item==cur->info)break;
124    cur=cur->rlink;
125    }
126    if(cur==head)
127    {
128     printf("key not found\n");
129     return head;
130     }
131     next=cur->rlink;
132     printf("enter towards right of %d=",item);
133     temp=getnode();
134     scanf("%d",&temp->info);
135     cur->rlink=temp;
136     temp->llink=cur;
137     next->llink=temp;
138     temp->rlink=next;
139     return head;
140    }
141    NODE search(NODE head,int item)
142    {
143     NODE temp,cur;
144      int flag=0;
145    if(head->rlink==head)
146    {
147    printf("list empty\n");
148    return head;
149    }
150    cur=head->rlink;
```

main.c

```
151     while(cur!=head)
152     {
153     if(item==cur->info)
154     {
155         flag=1;
156         break;
157     }
158     cur=cur->rlink;
159     }
160     if(cur==head)
161     printf("search unsuccessfull\n");
162     if(flag==1)
163     printf("search successfull\n");
164     }
165     NODE delete_all_key(int item,NODE head)
166     {
167         NODE prev,cur,next;
168         int count;
169         if(head->rlink==head)
170         {
171             printf("list empty\n");
172             return head;
173         }
174         count=0;
175         cur=head->rlink;
176         while(cur!=head)
177         {
178             if(item!=cur->info)
179             cur=cur->rlink;
```

main.c

```
210   {
211   printf("%d\n",temp->info);
212   temp=temp->rlink;
213   }
214   printf("\n");
215   }
216   int main()
217   {
218   NODE head,last;
219   int item, choice;
220   head=getnode();
221   head->rlink=head;
222   head->llink=head;
223
224   for(;;)
225   {
226     printf("\n1:insert front\n2:insert rear\n3:delete
      front\n4:delete rear\n5:insert left of key
      element\n6:insert right of key
      element\n7:search\n8:delete repeating
      occurances\n9:display\n10:exit\n");
227     printf("enter the choice\n");
228     scanf("%d", &choice);
229     switch(choice)
230     {
231       case 1: printf("enter the item at front end\n");
232         scanf("%d",&item);
233         last=dinsert_front(item,head);
234         break;
```

main.c

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

```
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
35

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
54

1:insert front
2:insert rear
3:delete front
```

```
7:search
8:delete repeating occurances
9:display
10:exit
enter the choice
8
enter element to be deleted
54
found at 1 positions and are deletedcontents of dq
35


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
8
enter element to be deleted
35
found at 1 positions and are deleteddq empty

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