

```

main.c
1  #include<stdio.h>
2  #include<conio.h>
3  #include<stdlib.h>
4  struct node
5  {
6      int info;
7      struct node *llink;
8      struct node *rlink;
9  };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE dinser_front(int item,NODE head)
27 {
28     NODE temp,cur;
29     temp=getnode();
30     temp->info=item;
31     cur=head->rlink;

```

```

31     cur=head->rlink;
32     head->rlink=temp;
33     temp->llink=head;
34     temp->rlink=cur;
35     cur->llink=temp;
36     return head;
37 }
38 NODE dinser_rear(int item,NODE head)
39 {
40     NODE temp,cur;
41     temp=getnode();
42     temp->info=item;
43     cur=head->llink;
44     head->llink=temp;
45     temp->rlink=head;
46     temp->llink=cur;
47     cur->rlink=temp;
48     return head;
49 }
50 NODE ddelete_front(NODE head)
51 {
52     NODE cur,next;
53     if(head->rlink==head)
54     {
55         printf("dq empty\n");
56         return head;
57     }
58     cur=head->rlink;
59     next=cur->rlink;
60     head->rlink=next;
61     next->llink=head;

```

```

61 next->llink=head;
62 printf("the node deleted is %d",cur->info);
63 freenode(cur);
64 return head;
65 }
66 NODE ddelete_rear(NODE head)
67 {
68     NODE cur,prev;
69     if(head->rlink==head)
70     {
71         printf("dq empty\n");
72         return head;
73     }
74     cur=head->llink;
75     prev=cur->llink;
76     head->llink=prev;
77     prev->rlink=head;
78     printf("the node deleted is %d",cur->info);
79     freenode(cur);
80     return head;
81 }
82
83 NODE insert_leftpos(int item,NODE head)
84 {
85     NODE temp,cur,prev;
86     if(head->rlink==head)
87     {
88         printf("list empty\n");
89         return head;
90     }
91     cur=head->rlink;
92     while(cur!=head)
93     {
94         if(item==cur->info)break;
95         cur=cur->rlink;
96     }
97     if(cur==head)
98     {
99         printf("key not found\n");
100         return head;
101     }
102     prev=cur->llink;
103     printf("enter towards left of %d=",item);
104     temp=getnode();
105     scanf("%d",&temp->info);
106     prev->rlink=temp;
107     temp->llink=prev;
108     cur->llink=temp;
109     temp->rlink=cur;
110     return head;
111 }
112
113 NODE insert_rightpos(int item,NODE head)
114 {
115     NODE temp,cur,next;
116     if(head->rlink==head)
117     {
118         printf("list empty\n");
119         return head;
120     }
121     cur=head->rlink;

```

```

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

```

```

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

```

```

181         else
182         {
183             count++;
184             prev=cur->llink;
185             next=cur->rlink;
186             prev->rlink=next;
187             next->llink=prev;
188             freenode(cur);
189             cur=next;
190         }
191     }
192     if(count==0)
193     printf("not found\n");
194     else{
195     printf("found at %d positions and are deleted",count);
196     return head;
197     }
198 }
199
200 void display(NODE head)
201 {
202     NODE temp;
203     if(head->rlink==head)
204     {
205     printf("dq empty\n");
206     return;
207     }
208     printf("contents of dq\n");
209     temp=head->rlink;
210     while(temp!=head)
211     {

```

```

210     while(temp!=head)
211     {
212     printf("%d",temp->info);
213     temp=temp->rlink;
214     }
215     printf("\n");
216     }
217 void main()
218 {
219     NODE head,last;
220     int item, choice;
221     head=getnode();
222     head->rlink=head;
223     head->llink=head;
224     //clrscr();
225     for(;;)
226     {
227         printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert left of k
228         printf("enter the choice\n");
229         scanf("%d",&choice);
230         switch(choice)
231         {
232             case 1: printf("enter the item at front end\n");
233                     scanf("%d",&item);
234                     last=dinsert_front(item,head);
235                     break;
236             case 2: printf("enter the item at rear end\n");
237                     scanf("%d",&item);
238                     last=dinsert_rear(item,head);
239                     break;
240             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: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.
```