```c
#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;
}
void freenode(NODE x)
{
    free(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;
```

```
    {
        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)
```

```
108     cur=head->rlink;
109     count=1;
110     while (cur!=head && cur->info!=key)
111     {
112         cur=cur->rlink;
113         count++;
114     }
115     if (cur==head)
116     {
117         printf("Search unsuccessfull\n");
118     }
119     else
120     {
121         printf("Key element found at the position %d\n",count);
122     }
123 }
124 NODE dinsert_leftpos(int item,NODE head)
125 {
126     NODE cur,prev,temp;
127     if (head->rlink==head)
128     {
129         printf("List is empty\n");
130         return head;
131     }
132     cur=head->rlink;
133     while (cur!=head)
134     {
135         if (cur->info==item)
136         {
137             break;
138         }
139         cur=cur->rlink;
140     }
141     if (cur==head)
142     {
143         printf("No such item found in the list\n");
144         return head;
145     }
```

```
142     {
143         printf("No such item found in the list\n");
144         return head;
145     }
146     prev=cur->llink;
147     temp=getnode();
148     temp->llink=NULL;
149     temp->rlink=NULL;
150     printf("Enter the item to be inserted at the left of the given item:\n");
151     scanf("%d",&temp->info);
152     prev->rlink=temp;
153     temp->llink=prev;
154     temp->rlink=cur;
155     cur->llink=temp;
156     return head;
157 }
158 NODE dinsert_rightpos(int item,NODE head)
159 {
160     NODE temp,cur,next;
161     if (head->rlink==head)
162     {
163         printf("List is empty\n");
164         return head;
165     }
166     cur=head->rlink;
167     while (cur!=head)
168     {
169         if (cur->info==item)
170         {
171             break;
172         }
173         cur=cur->rlink;
174     }
175     if (cur==head)
176     {
177         printf("No such item found in the list\n");
178         return head;
179     }
```

```
177         printf("No such item found in the list\n");
178         return head;
179     }
180     next=cur->rlink;
181     temp=getnode();
182     temp->llink=NULL;
183     temp->rlink=NULL;
184     printf("Enter the item to be inserted at the right of the given item:\n");
185     scanf("%d",&temp->info);
186     cur->rlink=temp;
187     temp->llink=cur;
188     next->llink=temp;
189     temp->rlink=next;
190     return head;
191 }
192 NODE ddelete_duplicates(int item,NODE head)
193 {
194     NODE prev,cur,next;
195     int count=0;
196     if (head->rlink==head)
197     {
198         printf("List is empty\n");
199         return head;
200     }
201     cur=head->rlink;
202     while (cur!=head)
203     {
204         if (cur->info!=item)
205         {
206             cur=cur->rlink;
207         }
208         else
209         {
210             count++;
211             if (count==1)
212             {
213                 cur=cur->rlink;
214                 continue;
```

```
213                 cur=cur->rlink;
214                 continue;
215             }
216             else
217             {
218                 prev=cur->llink;
219                 next=cur->rlink;
220                 prev->rlink=next;
221                 next->llink=prev;
222                 free(cur);
223                 cur=next;
224             }
225         }
226     }
227     if (count==0)
228     {
229         printf("No such item found in the list\n");
230     }
231     else
232     {
233         printf("All the duplicate elements of the given item are removed successfully\n");
234     }
235     return head;
236 }
237 NODE delete_all_key(int item,NODE head)
238 {
239 NODE prev,cur,next;
240 int count;
241    if(head->rlink==head)
242     {
243      printf("LE");
244      return head;
245      }
246 count=0;
247 cur=head->rlink;
248 while(cur!=head)
249 {
250   if(item!=cur->info)
```

```
  if(item!=cur->info)
  cur=cur->rlink;
  else
 {
  count++;
  prev=cur->llink;
  next=cur->rlink;
  prev->rlink=next;
  next->llink=prev;
  freenode(cur);
  cur=next;
  }
 }

if(count==0)
  printf("Key not found");
  else
 printf("Key found at %d positions and are deleted\n", count);

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
    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);
```

```
281     printf("Enter the choice\n");
282     scanf("%d",&choice);
283     switch(choice)
284     {
285         case 1: printf("Enter the item at front end:\n");
286                 scanf("%d",&item);
287                 head=dinsert_front(item,head);
288                 break;
289         case 2: printf("Enter the item at rear end:\n");
290                 scanf("%d",&item);
291                 head=dinsert_rear(item,head);
292                 break;
293         case 3:head=ddelete_front(head);
294                break;
295         case 4:head=ddelete_rear(head);
296                break;
297         case 5:ddisplay(head);
298                break;
299         case 6:printf("Enter the key element to be searched:\n");
300                scanf("%d",&key);
301                dsearch(key,head);
302                break;
303         case 7:printf("Enter the key element:\n");
304                scanf("%d",&key);
305                head=dinsert_leftpos(key,head);
306                break;
307         case 8:printf("Enter the key element:\n");
308                scanf("%d",&key);
309                head=dinsert_rightpos(key,head);
310                break;
311         case 9:printf("Enter the key element whose duplicates should be removed:\n");
312                scanf("%d",&key);
313                head=ddelete_duplicates(key,head);
314                break;
315         case 10:printf("Enter the key value\n");
316          scanf("%d",&item);
317          delete_all_key(item,head);
318          break;
```

```
288                 break;
289         case 2: printf("Enter the item at rear end:\n");
290                 scanf("%d",&item);
291                 head=dinsert_rear(item,head);
292                 break;
293         case 3:head=ddelete_front(head);
294                break;
295         case 4:head=ddelete_rear(head);
296                break;
297         case 5:ddisplay(head);
298                break;
299         case 6:printf("Enter the key element to be searched:\n");
300                scanf("%d",&key);
301                dsearch(key,head);
302                break;
303         case 7:printf("Enter the key element:\n");
304                scanf("%d",&key);
305                head=dinsert_leftpos(key,head);
306                break;
307         case 8:printf("Enter the key element:\n");
308                scanf("%d",&key);
309                head=dinsert_rightpos(key,head);
310                break;
311         case 9:printf("Enter the key element whose duplicates should be removed:\n");
312                scanf("%d",&key);
313                head=ddelete_duplicates(key,head);
314                break;
315         case 10:printf("Enter the key value\n");
316          scanf("%d",&item);
317          delete_all_key(item,head);
318          break;
319         case 11:exit(0);
320         default:printf("Invalid choice\n");
321         }
322     }
323     return 0;
324 }
325
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
2

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
3

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
3
2
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
6
Enter the key element to be searched:
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
Enter the choice
6
Enter the key element to be searched:
2
Key element found at the position 2

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
7
Enter the key element:
3
Enter the item to be inserted at the left of the given item:
6

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
1
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
8
Enter the key element:
1
Enter the item to be inserted at the right of the given item:
9

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
9
Enter the key element whose duplicates should be removed:
2
All the duplicate elements of the given item are removed successfully

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
6
Enter the key element to be searched:
2
Key element found at the position 3

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
1
9

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
10
Enter the key value
9
Key found at 1 positions and are deleted

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
11


-----------------
(program exited with code: 0)

Press any key to continue . . .
```