S.R. POOJA
1BM19CS135

LAB PROG - 9

```
#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;
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;
```

S.R. POOJA
1BM19CS135


```
return head; }
NODE ddelete_first(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;
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
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)
{
```

S.R.POOJA
1BM19CS135

```
printf("key not found\n");
return head; }
prev = cur->link;
printf("enter towards left of %d=", item);
temp = getnode();
scanf("%d", &temp->info);
prev->rlink = temp;
temp->llink = prev;
cur->llink = cur;
return head;
}
NODE insert_right_pos(int item, NODE head)
{ NODE temp, cur, next;
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;
} 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)
{
```

S.R.POOJA
1BM19CS135

```
NODE temp, cur;
int flag = 0;
if (head->rlink == head)
{ printf ("list empty\n");
return head;
} cur = head->rlink;
while (cur != head)
{ if (item == cur->info)
{ flag = 1;
  break;
} cur = cur->rlink;
}
if (cur == head)
printf (" search unsuccessful\n");
if (flag == 1)
printf ("search successful\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 {
count++;
 prev = cur->llink;
next = cur->rlink;
prev->rlink = next;
next->llink = prev;
free node(cur);
```

S.R. POOJA
1BM19CS135

```
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)
{
printf ("%d \n", temp -> info);
temp = temp -> rlink; }
printf ("\n");
} int main()
{
NODE head, last;
int item, choice;
head = getnode();
head -> rlink = head;
head -> llink = head;
for (;;)
{ 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
```

S.R.POOJA
1BM19CS135

```
occurances \n 9: display\n 10: 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);
  last = d insert_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);
  break;
case 4: last= ddelete_rear(head);
break;
case 5: printf ("enter the key element\n");
   scanf("%d", &item);
   last = insert_leftpos(item, head); break;
case 6: printf("enter the key element\n");
  scanf("%d", &item);
   search(head, item); last= insert_rightpos(item,head); break;
case 7: printf ("enter the search element\n");
   scanf ("%d", &item);
    search (head, item); break;
case 8: printf ("enter element to be deleted\n");
        scanf ("%d,&item);
        last = delete_all_key (item, head);
case 9: display (head);
        break;
   default: exit(0);
}
}
}
```

```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;
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();
```

```
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;
64  }
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;
```

```
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;
94  cur=cur->rlink;
95  }
96  if(cur==head)
97  {
98   printf("key not found\n");
99   return head;
100  }
101  prev=cur->llink;
102  printf("enter towards left of %d=",item);
103  temp=getnode();
104  scanf("%d",&temp->info);
105  prev->rlink=temp;
106  temp->llink=prev;
107  cur->llink=temp;
108  temp->rlink=cur;
109  return head;
110 }
111
112 NODE insert_rightpos(int item,NODE head)
113 {
114 NODE temp,cur,next;
115 if(head->rlink==head)
116 {
117 printf("list empty\n");
118 return head;
```

main.c

```
119   }
120   cur=head->rlink;
121   while(cur!=head)
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;
151   while(cur!=head)
152   {
153   if(item==cur->info)
154   {
155       flag=1;
156       break;
157   }
158   cur=cur->rlink;
```

```
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
        {
            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;
    }
}
```

```
198
199  void display(NODE head)
200  {
201  NODE temp;
202  if(head->rlink==head)
203  {
204  printf("dq empty\n");
205  return;
206  }
207  printf("contents of dq\n");
208  temp=head->rlink;
209  while(temp!=head)
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;
235      case 2: printf("enter the item at rear end\n");
```

```
{
  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);
    break;
  case 4: last=ddelete_rear(head);
    break;

  case 5:
        printf("enter the key element\n");
        scanf("%d",&item);
        last=insert_leftpos(item,head);
        break;
   case 6:
        printf("enter the key element\n");
        scanf("%d",&item);
        last=insert_rightpos(item,head);
        break;
  case 7:
            printf("enter the search element\n");
            scanf("%d",&item);
            search(head,item);
            break;
  case 8: printf("enter element to be deleted\n");
            scanf("%d",&item);
            last=delete_all_key(item,head);
  case 9: display(head);
    break;
  default:exit(0);
  }
}
}
```

```
enter the item at front end
12

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
21

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
3
the node deleted is 12
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
```

```
the node deleted is 21
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
5
enter the key element
17
list empty

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
dq empty

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
```