```
#include <stdio.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;
 cur=head->rlink;
 head->rlink=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;
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 delete_specified_value(int item, NODE head)
{
NODE prev, cur, next;
int count;
if(head->rlink == head)
{
printf("List is empty");
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 ("key found");
  else
  printf ("Key found at %d positions and are deleted\n",
          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");
  }
  void main()
  {
   NODE head, last;
   int item, choice;
    head = getnode();
    head->rlink = head;
```

```
head -> link = head;
for(;;)
{
printf("\n 1. IF \n 2. IR \n 3. DF \n 4. DR \n 5. Insert
        left position \n 6. Delete specified value \n 7. Display \n
        8. Exit \n");
printf("Enter your choice \n");
scanf("%d", &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);
        break;
case 4: last = ddelete_rear(head);
        break;
case 5: printf("Enter the key item \n");
        scanf("%d", &item);
        head = insert_leftpos(item, head);
        break;
case 6: printf("Enter the key item \n");
        scanf("%d", &item);
        head = delete_specified_value(item, head);
        break;
case 7: display(head);
        break;
default: exit(0); } } }
```

```c
#include <stdio.h>
#include <stdlib.h>
struct node{
 int info;
 struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
 NODE x;
 x=(NODE)malloc(sizeof(NODE));//
 if(x==NULL)
 {
  printf("memory full \n");
  exit(0);
 }
 return x;
}

void freenode(NODE x)
{
 free(x);
}

NODE insert_front(NODE first,int item)
{
 NODE temp = getnode();
 temp->info = item;
 temp->link = NULL;
 if(first == NULL)
 return temp;
 temp->link=first;
 return temp;
}

NODE delete_front(NODE first)
{
 NODE temp;
 if(first == NULL)
 {
  printf("List is empty\n");
  return first;
 }
 printf("Item deleted %d",(first->info));
 temp = first;
 temp=temp->link;
 free(first);
 return temp;
}

NODE insert_rear(NODE first,int item)
{
 NODE temp = getnode(),cur;
```

```
 NODE temp = getnode(),cur;
 temp->info=item;
 temp->link = NULL;
 if(first==NULL)
 return temp;
 cur=first;
 while(cur->link!=NULL)
 cur=cur->link;
 cur->link=temp;
 return first;
}

NODE delete_rear(NODE first)
{
 NODE cur=first,prev=NULL;
 if(first==NULL)
 {
  printf("List empty\n");
  return NULL;
 }
 if(first->link==NULL)
 {
  printf("item deleted is
%d\n",first->info);
  free(first);
  return NULL;
 }
 while(cur->link!=NULL)
 {
  prev=cur;
  cur=cur->link;
 }
 printf("Item deleted is %d\n",(cur->info));
 free(cur);
 prev->link=NULL;
 return first;
}

void display(NODE first)
{
 if(first==NULL)
 {
  printf("List is empty\n");
  return;
 }
 printf("Elements of the list are : \n");
 for(NODE i=first;i!=NULL;i=i->link)
 printf("%d\n",i->info);
}
NODE insert_pos(NODE first,int item,int pos)
{
 int c=1;
 NODE temp = getnode(),cur,prev;
```

```
105   temp->info=item;
106   if(pos==1)
107   {
108     temp->link=first;
109     return temp;
110   }
111   cur=first;
112   prev=NULL;
113   while(cur!=NULL)
114   {
115     if(pos==c)
116     {
117       prev->link=temp;
118       temp->link=cur;
119       return first;
120     }
121     c++;
122     prev=cur;
123     cur=cur->link;
124   }
125   printf("Invalid position\n");
126   return first;
127 }
128
129 NODE delete_pos(NODE first,int pos)
130 {
131   NODE cur,prev;
132   int c=1;
133   if(first==NULL||pos<0)
134   {
135     printf("Invalid position\n");
136     return NULL;
137   }
138   if(pos==1)
139   {
140     free(first);
141     return NULL;
142   }
143   cur=first;
144   prev=NULL;
145   while(cur!=NULL)
146   {
147     if(c==pos)
148     {
149       printf("Element deleted is
      %d",cur->info);
150       prev->link=cur->link;
151       free(cur);
152       return first;
153     }
154     prev=cur;
155     cur=cur->link;
156     c++;
```

```
157   }
158   printf("Element not found\n");
159   return first;
160 }
161 NODE delete_key(NODE first,int key)
162 {
163   NODE prev,cur;
164   if(first==NULL)
165   {
166     printf("List is empty\n");
167     return NULL;
168   }
169   if(key==first->info)
170   {
171     cur=first;
172     first=first->link;
173     free(cur);
174     printf("Element deleted successfully\n");
175     return first;
176   }
177   cur=first;
178   prev=NULL;
179   while(cur!=NULL)
180   {
181     if(key==cur->info)
182     {
183       printf("Item deleted successfully\n");
184       prev->link=cur->link;
185       free(cur);
186       return first;
187     }
188     prev=cur;
189     cur=cur->link;
190   }
191   if(cur==NULL)
192   printf("Element not found \n");
193   return first;
194 }
195
196 NODE reverse_list(NODE first)
197 {
198   NODE cur,temp;
199   cur = NULL;
200   while(first!=NULL)
201   {
202     temp = first;
203     first=first->link;
204     temp->link=cur;
205     cur=temp;
206   }
207   printf("List has been reversed
    successfully\n");
208   return cur;
```

```
208   return cur;
209 }
210
211
212 int main()
213 {
214   int item,ch,pos;
215   NODE first=NULL;
216   for(;;)
217   {
218     printf("\n1.Insert front\n2.Delete
    front\n3.Insert rear\n4.delete
    rear\n5.Insert_pos\n6.Delete_pos\n7.Delete
    key\n8.Display\n9.Reverse\n10.Exit\n");
219     scanf("%d",&ch);
220     switch(ch)
221     {
222       case 1:
223         printf("Enter element to be
    inserted\n");
224         scanf("%d",&item);
225         first = insert_front(first,item);
226         break;
227       case 2:
228         first = delete_front(first);
229         break;
230       case 3:
231         printf("Enter element to be
    inserted\n");
232         scanf("%d",&item);
233         first = insert_rear(first,item);
234         break;
235       case 4:
236         first = delete_rear(first);
237         break;
238       case 5:
239         printf("Enter element to be
    inserted\n");
240         scanf("%d",&item);
241         printf("Enter position\n");
242         scanf("%d",&pos);
243         first = insert_pos(first,item,pos);
244         break;
245       case 6:
246         printf("Enter position\n");
```

```
247         scanf("%d",&pos);
248         first = delete_pos(first,pos);
249         break;
250       case 7:
251         printf("Enter element to be
    deleted\n");
252         scanf("%d",&item);
253         first = delete_key(first,item);
254         break;
255       case 8:
256         display(first);
257         break;
258       case 9:
259         first=reverse_list(first);
260         break;
261       default:return 0;
262     }
263   }
264 }
```

```
1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
1
Enter element to be inserted
10

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
3
Enter element to be inserted
20

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
5
Enter element to be inserted
30
Enter position
3

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
```

```
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
8
Elements of the list are :
10
20
30
40

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
2
Item deleted 10
1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
4
Item deleted is 40

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
```

```
7
Enter element to be deleted
20
Element deleted successfully

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
8
Elements of the list are :
30

1.Insert front
2.Delete front
3.Insert rear
4.delete rear
5.Insert_pos
6.Delete_pos
7.Delete key
8.Display
9.Reverse
10.Exit
```