```c
#include<stdio.h>
#include<process.h>
int n,i;
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE a,b;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("Memory is full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
```

```
43      temp->link=NULL;
44      if(first==NULL)
45      return temp;
46      cur=first;
47      while(cur->link!=NULL)
48      cur=cur->link;
49      cur->link=temp;
50      return first;
51      }
52      NODE insert_pos(int item,int pos,NODE first)
53      {
54      NODE temp,cur,prev;
55      int count;
56      temp=getnode();
57      temp->info=item;
58      temp->link=NULL;
59      if(first==NULL&&pos==1)
60      {
61      return temp;
62      }
63      if(first==NULL)
64      {
65      printf("invalid position\n");
66      return first;
67      }
68      if(pos==1)
69      {
70      temp->link=first;
71      first=temp;
72      return temp;
73      }
74      count=1;
75      prev=NULL;
76      cur=first;
77      while(cur!=NULL&&count!=pos)
78      {
79      prev=cur;
80      cur=cur->link;
81      count++;
82      }
83      if(count==pos)
84      {
```

```
86      prev->link=temp;
87      temp->link=cur;
88      return first;
89      }
90      printf("invalid position\n");
91      return first;
92      }
93      NODE delete_front(NODE first)
94      {
95      NODE temp;
96      if(first==NULL)
97      {
98      printf("CANNOT DELETE AS LIST IS EMPTY\n");
99      return first;
100     }
101     temp=first;
102     temp=temp->link;
103     printf("ITEM DELETED AT FRONT END=%d\n",first->info);
104     free(first);
105     return temp;
106     }
107     NODE delete_rear(NODE first)
108     {
109     NODE cur,prev;
110     if(first==NULL)
111     {
112     printf("CANNOT DELETE AS LIST IS EMPTY\n");
113     return first;
114     }
115     if(first->link==NULL)
116     {
117     printf("ITEM DELETED=%d\n",first->info);
118     free(first);
119     return NULL;
120     }
121     prev=NULL;
122     cur=first;
123     while(cur->link!=NULL)
124     {
125     prev=cur;
126     cur=cur->link;
127     }
```

```
printf("ITEM DELETED AT REAR END=%d\n",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE delete_pos(int pos,NODE first)
{
NODE cur;
NODE prev;
int count,flag=0;
if(first==NULL || pos<0)
{
printf("invalid position\n");
return NULL;
}
if(pos==1)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
{
if(count==pos)
{
flag=1;
break;
}
count++;
prev=cur;
cur=cur->link;
}
if(flag==0)
{
printf("invalid position\n");
return first;
}
printf("ITEM DELETED AT POSITION %d is %d\n",pos,cur->info);
```

```
prev->link=cur->link;
freenode(cur);
return first;
}
NODE sort_asc(NODE first)
{
  int tmp;
  NODE cur,next;
  cur=first;
  next=NULL;
  if(first==NULL){
  printf("List is empty\n");
  return;
  }
  while(cur!=NULL)
  {
   next=cur->link;
    while(next!=NULL)
   {
   if(cur->info>next->info)
   {
    tmp=cur->info;
    cur->info=next->info;
    next->info=tmp;
   }
   next=next->link;
  }
  cur=cur->link;
  }
  return first;
}
NODE sort_desc(NODE first)
{
  int tmp;
  NODE cur,next;
  cur=first;
  next=NULL;
  if(first==NULL){
  printf("List is empty\n");
  return;
  }
  while(cur!=NULL)
```

```
212     {
213      next=cur->link;
214       while(next!=NULL)
215      {
216      if(cur->info<next->info)
217      {
218       tmp=cur->info;
219       cur->info=next->info;
220       next->info=tmp;
221      }
222      next=next->link;
223     }
224     cur=cur->link;
225     }
226     return first;
227   }
228   NODE reverse(NODE first)
229  {
230     NODE cur,temp;
231     cur=NULL;
232     while(first!=NULL)
233     {
234       temp=first;
235       first=first->link;
236       temp->link=cur;
237       cur=temp;
238     }
239     return cur;
240   }
241   NODE concat(NODE first,NODE second)
242  {
243    NODE cur;
244    if(first==NULL)
245     return second;
246    if(second==NULL)
247     return first;
248    cur=first;
249    while(cur->link!=NULL)
250     cur=cur->link;
251     cur->link=second;
252     return first;
253   }
```

```
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("1.Insert_front\n2.Insert_rear\n3.Insert at given Position\n4.Delete Front\n5.Delete Rear\n6.Delete at a given position\n7.Display the list\n8.Sort in Ascending\n9.Sort in Descending
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at front-end\n");
       scanf("%d",&item);
       first=insert_front(first,item);
       break;
case 2:printf("enter the item at rear-end\n");
       scanf("%d",&item);
       first=insert_rear(first,item);
       break;
case 3:printf("enter the item to be inserted at given position\n");
       scanf("%d",&item);
       printf("enter the position\n");
       scanf("%d",&pos);
       first=insert_pos(item,pos,first);
       break;
case 4:first=delete_front(first);
       break;
case 5:first=delete_rear(first);
       break;
case 6:printf("Enter the position\n");
       scanf("%d",&pos);
       first=delete_pos(pos,first);
```

```
\n4.Delete Front\n5.Delete Rear\n6.Delete at a given position\n7.Display the list\n8.Sort in Ascending\n9.Sort in Descending\n10.Reverse\n11.Concat\n12.Exit\n");
```

```
);
```

```
case 6:printf("Enter the position\n");
       scanf("%d",&pos);
       first=delete_pos(pos,first);
       break;
case 7:display(first);
       break;
case 8:first=sort_asc(first);
       break;
case 9:first=sort_desc(first);
       break;
case 10:first=reverse(first);
        display(first);
        break;
case 11:printf("Enter number of nodes in List2\n");
        scanf("%d",&n);
        a=NULL;
        for(i=0;i<n;i++)
        {
         printf("Enter Item:\n");
         scanf("%d",&item);
         a=insert_rear(a,item);
        }
        /*printf("Enter number of nodes in List3\n");
        scanf("%d",&n);
        b=NULL;
        for(i=0;i<n;i++)
        {
         printf("Enter Item:\n");
         scanf("%d",&item);
         b=insert_rear(b,item);
        }*/
        first=concat(first,a);
        display(first);
        break;
default:exit(0);
        break;
}
}
}
```

"C:\Users\Neha Chadaga\Desktop\LinkedList3.exe"

```
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
1
enter the item at front-end
23
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
1
enter the item at front-end
45
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
2
enter the item at rear-end
78
1.Insert_front
2.Insert_rear
```

```
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
2
enter the item at rear-end
90
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
3
enter the item to be inserted at given position
2
enter the position
2
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
6
Enter the position
5
```

```
ITEM DELETED AT POSITION 5 is 90
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
7
45
2
23
78
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
8
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
7
```

```
2
23
45
78
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
9
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
7
78
45
23
2
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
```

```
enter the choice
10
2
23
45
78
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
11
Enter number of nodes in List2
4
Enter Item:
12
Enter Item:
13
Enter Item:
14
Enter Item:
15
2
23
45
78
12
13
14
15
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
```

```
Enter Item:
15
2
23
45
78
12
13
14
15
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Sort in Ascending
9.Sort in Descending
10.Reverse
11.Concat
12.Exit
enter the choice
12

Process returned 0 (0x0)   execution time : 86.625 s
Press any key to continue.
```