```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct node
{
  int info;
  struct node *link;
};
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 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;
```

```
36 return first;
37 }
38 NODE IF(NODE second,int item)
39 {
40 NODE temp;
41 temp=getnode();
42 temp->info=item;
43 temp->link=NULL;
44 if(second==NULL)
45 return temp;
46 temp->link=second;
47 second=temp;
48 return second;
49 }
50 NODE delete_front(NODE first)
51 {
52 NODE temp;
53 if(first==NULL)
54 {
55 printf("list is empty cannot delete\n");
56 return first;
57 }
58 temp=first;
59 temp=temp->link;
60 printf("item deleted at front-end is=%d\n",first-
   >info);
61 free(first);
62 return temp;
63 }
64 NODE insert_rear(NODE first,int item)
65 {
66 NODE temp,cur;
67 temp=getnode();
68 temp->info=item;
69 temp->link=NULL;
70 if(first==NULL)
```

```
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}
NODE IR(NODE second,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(second==NULL)
 return temp;
cur=second;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return second;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
```

```
107 cur=first;
108 while(cur->link!=NULL)
109 {
110 prev=cur;
111 cur=cur->link;
112 }
113 printf("iten deleted at rear-end is %d",cur->info);
114 free(cur);
115 prev->link=NULL;
116 return first;
117 }
118 NODE insert_pos(int item,int pos,NODE first)
119 {
120 NODE temp;
121 NODE prev,cur;
122 int count;
123 temp=getnode();
124 temp->info=item;
125 temp->link=NULL;
126 if(first==NULL && pos==1)
127 return temp;
128 if(first==NULL)
129 {
130  printf("invalid pos\n");
131  return first;
132 }
133 if(pos==1)
134 {
135 temp->link=first;
136 return temp;
137 }
138 count=1;
139 prev=NULL;
140 cur=first;
141 while(cur!=NULL && count!=pos)
142 {
```

```
 prev=cur;
 cur=cur->link;
 count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid Position \n");
return first;
}
NODE delete_pos(int pos, NODE first)
{
NODE cur;
NODE prev;
int count;
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)
```

```
179 break;  //if found
180 prev=cur;
181 cur=cur->link;
182 count++;
183 }
184 if(count!=pos)
185 {
186   printf("invalid position\n");
187    return first;
188 }
189 if(count!=pos)
190 {
191   printf("invalid position specified\n");
192   return first;
193 }
194
195 prev->link=cur->link;
196  freenode(cur);
197 return first;
198 }
199 NODE reverse(NODE first)
200  {
201    NODE cur,temp;
202    cur=NULL;
203    while(first!=NULL)
204    {
205       temp=first;
206       first=first->link;
207       temp->link=cur;
208       cur=temp;
209    }
210    return cur;
211 }
212 NODE asc(NODE first)
213 {
214    NODE prev=first;
```

```
215     NODE cur=NULL;
216           int temp;
217
218 if(first== NULL) {
219       return 0;
220       }
221 else {
222      while(prev!= NULL) {
223
224       cur = prev->link;
225
226       while(cur!= NULL) {
227
228        if(prev->info > cur->info) {
229       temp = prev->info;
230       prev->info = cur->info;
231       cur->info = temp;
232        }
233        cur = cur->link;
234       }
235       prev= prev->link;
236             }
237          }
238          return first;
239        }
240 NODE des(NODE first)
241 {
242     NODE prev=first;
243     NODE cur=NULL;
244           int temp;
245
246 if(first==NULL) {
247     return 0;
248      }
249     else {
250       while(prev!= NULL) {
```

```
      cur = prev->link;

       while(cur!= NULL) {

       if(prev->info < cur->info) {
      temp = prev->info;
      prev->info = cur->info;
      cur->info = temp;
         }
         cur = cur->link;
         }
         prev= prev->link;
          }
          }
          return first;
       }
NODE concate(NODE first,NODE second)
{
   NODE cur;
   if(first==NULL)
   return second;
   if(second==NULL)
   return first;
   cur=first;
   while(cur->link!=NULL)
   {
      cur=cur->link;

   }
   cur->link=second;
   return first;
}

void display(NODE first)
{
```

```
287  NODE temp;
288  if(first==NULL)
289  printf("list empty cannot display items\n");
290  for(temp=first;temp!=NULL;temp=temp->link)
291   {
292   printf("%d\n",temp->info);
293   }
294 }
295 void main()
296 {
297 int item,choice,pos,element,option,choice2,
     item1,num;
298 NODE first=NULL;
299 NODE second=NULL;
300
301 for(;;)
302 {
303 printf("\n 1:Insert_front\n 2:Delete_front\n
     3:Insert_rear\n 4:Delete_rear\n
     5:random_position\n 6:reverse\n 7:sort\n 8.
     concate\n 9:display_list\n 10:Exit\n");
304 printf("enter the choice\n");
305 scanf("%d",&choice);
306 switch(choice)
307  {
308   case 1:printf("enter the item at front-end\n");
309      scanf("%d",&item);
310      first=insert_front(first,item);
311      break;
312   case 2:first=delete_front(first);
313      break;
314   case 3:printf("enter the item at rear-end\n");
315      scanf("%d",&item);
316      first=insert_rear(first,item);
317      break;
318   case 4:first=delete_rear(first);
```

```
318   case 4:first=delete_rear(first);
319      break;
320   case 5:
321   printf("press 1 to insert or 2 to delete at any
      desired position \n");
322   scanf("%d",&element);
323   if(element==1){
324         printf("enter the position to insert \n");
325         scanf("%d",&pos);
326         printf("enter the item to insert \n");
327         scanf("%d",&item);
328         first=insert_pos(item,pos,first);
329   }
330   if(element==2){
331         printf("enter the position to delete \n");
332         scanf("%d",&pos);
333         first=delete_pos(pos,first);
334   }
335         break;
336    case 6:
337        first=reverse(first);
338        break;
339    case 7:
340        printf("press 1 for ascending sort and 2
      for descending sort:\n");
341        scanf("%d",&option);
342        if(option==1)
343        first=asc(first);
344        if(option==2)
345        first=des(first);
346        break;
347    case 8:
348        printf("create a second list\n");
349        printf("enter the number of elements in
      second list\n");
350
```

```
348         printf("create a second list\n");
349         printf("enter the number of elements in
    second list\n");
350
351         scanf("%d",&num);
352          for(int i=1;i<=num;i++){
353         printf("\n press 1 to insert front and 2 to
    insert rear \n");
354         scanf("%d",&choice2);
355
356            if(choice2==1){
357            printf("enter the item at front-end\n");
358            scanf("%d",&item1);
359           second=IF(second,item1);
360            }
361
362           if(choice2==2){
363           printf("enter the item at rear-end\n");
364           scanf("%d",&item1);
365           second=IR(second,item1);
366           }
367          }
368
369         first=concate(first,second);
370         break;
371
372   case 9:display(first);
373      break;
374   default:exit(0);
375      break;
376   }
377  }
378  getch();
379  }
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
1
enter the item at front-end
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
3
enter the item at rear-end
3

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
9
2
3

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
2
item deleted at front-end is=2

 1:Insert_front
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
1
enter the item at front-end
6

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
4
iten deleted at rear-end is 3
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
3
enter the item at rear-end
8

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
1
enter the item at front-end
2

 1:Insert_front
 2:Delete_front
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
5
press 1 to insert or 2 to delete at any desired position
1
enter the position to insert
3
enter the item to insert
4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
9
2
6
4
8

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
6

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
```

```
enter the choice
9
8
4
6
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
7
press 1 for ascending sort and 2 for descending sort:
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
9
2
4
6
8

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
8
create a second list
enter the number of elements in second list
3

 press 1 to insert front and 2 to insert rear
1
enter the item at front-end
1

 press 1 to insert front and 2 to insert rear
```

```
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
8
create a second list
enter the number of elements in second list
3

 press 1 to insert front and 2 to insert rear
1
enter the item at front-end
1

 press 1 to insert front and 2 to insert rear
1
enter the item at front-end
2

 press 1 to insert front and 2 to insert rear
1
enter the item at front-end
3

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
9
2
4
6
8
3
2
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
 9:display_list
 10:Exit
enter the choice
```