main.c

```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(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;
```

main.c

```
30    temp->link=NULL;
31    if(first==NULL)
32    return temp;
33    temp->link=first;
34    first=temp;
35    return first;
36    }
37    NODE IF(NODE second,int item)
38    {
39    NODE temp;
40    temp=getnode();
41    temp->info=item;
42    temp->link=NULL;
43    if(second==NULL)
44    return temp;
45    temp->link=second;
46    second=temp;
47    return second;
48    }
49    NODE delete_front(NODE first)
50    {
51    NODE temp;
52    if(first==NULL)
53    {
54    printf("list is empty cannot delete\n");
55    return first;
56    }
57    temp=first;
```

main.c

```
57    temp=first;
58    temp=temp->link;
59    printf("item deleted at front-end is=%d\n",first->info);
60    free(first);
61    return temp;
62    }
63    NODE insert_rear(NODE first,int item)
64    {
65    NODE temp,cur;
66    temp=getnode();
67    temp->info=item;
68    temp->link=NULL;
69    if(first==NULL)
70     return temp;
71    cur=first;
72    while(cur->link!=NULL)
73     cur=cur->link;
74    cur->link=temp;
75    return first;
76    }
77    NODE IR(NODE second,int item)
78    {
79    NODE temp,cur;
80    temp=getnode();
81    temp->info=item;
82    temp->link=NULL;
83    if(second==NULL)
84     return temp;
85    cur=second;
```

main.c

```
84      return temp;
85      cur=second;
86      while(cur->link!=NULL)
87      cur=cur->link;
88      cur->link=temp;
89      return second;
90      }
91      NODE delete_rear(NODE first)
92      {
93      NODE cur,prev;
94      if(first==NULL)
95      {
96      printf("list is empty cannot delete\n");
97      return first;
98      }
99      if(first->link==NULL)
100     {
101     printf("item deleted is %d\n",first->info);
102     free(first);
103     return NULL;
104     }
105     prev=NULL;
106     cur=first;
107     while(cur->link!=NULL)
108     {
109     prev=cur;
110     cur=cur->link;
111     }
112     printf("item deleted at rear-end is %d",cur->info);
```

main.c

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

main.c

```
138    prev=NULL;
139    cur=first;
140    while(cur!=NULL && count!=pos)
141    {
142     prev=cur;
143     cur=cur->link;
144     count++;
145    }
146    if(count==pos)
147    {
148    prev->link=temp;
149    temp->link=cur;
150    return first;
151    }
152    printf("Invalid Position \n");
153    return first;
154    }
155    NODE delete_pos(int pos, NODE first)
156    {
157    NODE cur;
158    NODE prev;
159    int count;
160    if(first==NULL || pos<=0)
161    {
162    printf("invalid position \n");
163    return NULL;
164    }
165    if (pos==1)
166    {
```

```
166    {
167    cur=first;
168    first=first->link;
169    freenode(cur);
170    return first;
171    }
172    prev=NULL;
173    cur=first;
174    count=1;
175    while(cur!=NULL)
176    {
177    if(count==pos)
178    break;  //if found
179    prev=cur;
180    cur=cur->link;
181    count++;
182    }
183    if(count!=pos)
184    {
185      printf("invalid position\n");
186       return first;
187    }
188    if(count!=pos)
189    {
190      printf("invalid position specified\n");
191      return first;
192    }
193
194    prev->link=cur->link;
```

main.c

```
194     prev->link=cur->link;
195     freenode(cur);
196     return first;
197     }
198     NODE reverse(NODE first)
199       {
200         NODE cur,temp;
201         cur=NULL;
202         while(first!=NULL)
203         {
204             temp=first;
205             first=first->link;
206             temp->link=cur;
207             cur=temp;
208         }
209         return cur;
210     }
211     NODE asc(NODE first)
212     {
213         NODE prev=first;
214         NODE cur=NULL;
215                 int temp;
216
217     if(first== NULL) {
218           return 0;
219           }
220     else {
221         while(prev!= NULL) {
222
```

main.c

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

main.c

```
250
251     cur = prev->link;
252
253     while(cur!= NULL) {
254
255       if(prev->info < cur->info) {
256     temp = prev->info;
257     prev->info = cur->info;
258     cur->info = temp;
259         }
260         cur = cur->link;
261         }
262         prev= prev->link;
263          }
264          }
265          return first;
266       }
267 NODE concate(NODE first,NODE second)
268 {
269     NODE cur;
270     if(first==NULL)
271     return second;
272     if(second==NULL)
273     return first;
274     cur=first;
275     while(cur->link!=NULL)
276     {
277         cur=cur->link;
```

main.c

```
277             cur=cur->link;
278
279         }
280         cur->link=second;
281         return first;
282     }
283
284     void display(NODE first)
285     {
286      NODE temp;
287      if(first==NULL)
288      printf("list empty cannot display items\n");
289      for(temp=first;temp!=NULL;temp=temp->link)
290       {
291       printf("%d\n",temp->info);
292       }
293     }
294     void main()
295     {
296     int item,choice,pos,element,option,choice2,item1,num;
297     NODE first=NULL;
298     NODE second=NULL;
299
300     for(;;)
301     {
302     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");
```

main.c

```
10:Exit\n");
303  printf("enter the choice\n");
304  scanf("%d",&choice);
305  switch(choice)
306   {
307    case 1:printf("enter the item at front-end\n");
308     scanf("%d",&item);
309     first=insert_front(first,item);
310     break;
311    case 2:first=delete_front(first);
312     break;
313    case 3:printf("enter the item at rear-end\n");
314     scanf("%d",&item);
315     first=insert_rear(first,item);
316     break;
317    case 4:first=delete_rear(first);
318     break;
319   case 5:
320   printf("press 1 to insert or 2 to delete at any desired
     position \n");
321   scanf("%d",&element);
322   if(element==1){
323          printf("enter the position to insert \n");
324      scanf("%d",&pos);
325      printf("enter the item to insert \n");
326      scanf("%d",&item);
327      first=insert_pos(item,pos,first);
328   }
```

main.c

```
329     if(element==2){
330         printf("enter the position to delete \n");
331         scanf("%d",&pos);
332         first=delete_pos(pos,first);
333     }
334         break;
335     case 6:
336         first=reverse(first);
337         break;
338     case 7:
339         printf("press 1 for ascending sort and 2 for
            descending sort:\n");
340         scanf("%d",&option);
341         if(option==1)
342         first=asc(first);
343         if(option==2)
344         first=des(first);
345         break;
346     case 8:
347         printf("create a second list\n");
348         printf("enter the number of elements in second
            list\n");
349
350         scanf("%d",&num);
351          for(int i=1;i<=num;i++){
352         printf("\n press 1 to insert front and 2 to insert
            rear \n");
353         scanf("%d",&choice2);
354
```

main.c

```
353             scanf("%d",&choice2);
354
355                 if(choice2==1){
356                 printf("enter the item at front-end\n");
357               scanf("%d",&item1);
358                 second=IF(second,item1);
359                  }
360
361                 if(choice2==2){
362                 printf("enter the item at rear-end\n");
363               scanf("%d",&item1);
364                 second=IR(second,item1);
365                 }
366               }
367
368             first=concate(first,second);
369             break;
370
371     case 9:display(first);
372      break;
373    default:exit(0);
374      break;
375    }
376   }
377   getch();
378   }
```

```
 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
34

 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
23

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
```

```
 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
2
item deleted at front-end is=34

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:random_position
 6:reverse
 7:sort
 8.concate
```