

main.c

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<conio.h>
4  struct node
5  {
6      int info;
7      struct node*link;
8
9  };
10 typedef struct node*NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("memory full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert_front(NODE first,int item)
27 {
28     NODE temp;
29     temp=getnode();
30     temp->info=item;
31     temp->link=NULL;
32     if(first==NULL)
33         return temp;
34     temp->link=first;
35     first=temp;
36     return first;

```

```
37 }
38 NODE delete_front(NODE first)
39 {
40     NODE temp;
41     if(first==NULL)
42     {
43         printf("list is empty cannot delete\n");
44         return first;
45     }
46     temp=first;
47     temp=temp->link;
48     printf("item deleted at front end is=%d \n",first->info);
49     free(first);
50     return temp;
51 }
52 NODE insert_rear(NODE first,int item)
53 {
54     NODE temp,cur;
55     temp=getnode();
56     temp->info=item;
57     temp->link=NULL;
58     if(first==NULL)
59         return temp;
60     cur=first;
61     while(cur->link!=NULL)
62         cur=cur->link;
63     cur->link=temp;
64     return first;
65 }
66 NODE delete_rear(NODE first)
67 {
68     NODE cur,prev;
69     if(first==NULL)
70     {
71         printf("list is empty cannot delete\n");
72         return first;
73     }
```

```
69 if(first==NULL)
70 {
71     printf("list is empty cannot delete\n");
72     return first;
73 }
74 if(first->link==NULL)
75 {
76     printf("item deleted is %d\n",first->info);
77     free(first);
78     return NULL;
79 }
80 prev=NULL;
81 cur=first;
82 while(cur->link!=NULL)
83 {
84     prev=cur;
85     cur=cur->link;
86 }
87 printf("item deleted at rear end is %d",cur->info);
88 free(cur);
89 prev->link=NULL;
90 return first;
91 }
92
93 NODE delete_pos(int pos,NODE first)
94 {
95     NODE prev,cur;
96     int count;
97     if (first==NULL || pos<=0)
98     {
99         printf("Invalid position\n");
100         return NULL;
101     }
102     if (pos==1)
103     {
104         cur=first;
```

```
138 - {
139     return temp;
140 }
141 if (first==NULL)
142 - {
143     printf("Invalid position\n");
144     return NULL;
145 }
146 if (pos==1)
147 - {
148     temp->link=first;
149     return temp;
150 }
151 count=1;
152 prev=NULL;
153 cur=first;
154 while (cur!=NULL && count!=pos)
155 - {
156     prev=cur;
157     cur=cur->link;
158     count++;
159 }
160 if (count==pos)
161 - {
162     prev->link=temp;
163     temp->link=cur;
164     return first;
165 }
166 printf("Invalid position\n");
167 return first;
168 }
169
170 void display(NODE first)
171 - {
172     NODE temp;
173     if (first==NULL)
```



```
102     if (pos==1)
103     {
104         cur=first;
105         first=first->link;
106         freenode(cur);
107         return first;
108     }
109     prev=NULL;
110     cur=first;
111     count=1;
112     while (cur!=NULL)
113     {
114         if (count==pos)
115         {
116             break;
117         }
118         prev=cur;
119         cur=cur->link;count++;
120     }
121     if (count!=pos)
122     {
123         printf("Invalid position\n");
124         return first;
125     }
126     prev->link=cur->link;
127     freenode(cur);
128     return first;
129 }
130 NODE insert_pos(int item,int pos,NODE first)
131 {
132     NODE temp,cur,prev;
133     int count;
134     temp=getnode();
135     temp->info=item;
136     temp->link=NULL;
137     if (first==NULL && pos==1)
```

main.c

```
170 void display(NODE *first)
171 {
172     NODE temp;
173     if(first==NULL)
174         printf("list empty cannot display items\n");
175     for(temp=first;temp!=NULL;temp=temp->link)
176     {
177         printf("%d\n",temp->info);
178     }
179 }
180 void main()
181 {
182     int item,choice,pos;
183     NODE first=NULL;
184     for(;;)
185     {
186         printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Delete at specified position\n 6:Insert at specified position\n 7:Display\n");
187         printf("enter the choice\n");
188         scanf("%d",&choice);
189         printf("-----\n");
190         switch(choice)
191         {
192             case 1:printf("enter the item at front end\n");
193                     scanf("%d",&item);
194                     first=insert_front(first,item);
195                     break;
196             case 2:first=delete_front(first);
197                     break;
198             case 3:printf("enter the item at rear end\n");
199                     scanf("%d",&item);
200                     first=insert_rear(first,item);
201                     break;
202             case 4:first=delete_rear(first);
203                     break;
204             case 5:printf("Enter the position:\n");
205                     scanf("%d",&pos);
```

```
184 for(;;)
185 {
186     printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Delete at specified pos\n");
187     printf("enter the choice\n");
188     scanf("%d",&choice);
189     printf("-----\n");
190     switch(choice)
191     {
192         case 1:printf("enter the item at front end\n");
193             scanf("%d",&item);
194             first=insert_front(first,item);
195             break;
196         case 2:first=delete_front(first);
197             break;
198         case 3:printf("enter the item at rear end\n");
199             scanf("%d",&item);
200             first=insert_rear(first,item);
201             break;
202         case 4:first=delete_rear(first);
203             break;
204         case 5:printf("Enter the position:\n");
205             scanf("%d",&pos);
206             first=delete_pos(pos,first);
207             break;
208         case 6:printf("Enter the item and the position:\n");
209             scanf("%d%d",&item,&pos);
210             first=insert_pos(item,pos,first);
211             break;
212         case 7:display(first);
213             break;
214         default:exit(0);
215         break;
216     }
217 }
218 }
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
```

enter the choice

1

enter the item at front end

23

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
```

enter the choice

1

enter the item at front end

12

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
```



```
4:Delete_rear
5:Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
enter the choice
6
-----
Enter the item and the position:
67
2

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
enter the choice
7
-----
12
67
23

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
```

```
7:Display_list
8:EXIT
enter the choice
7
-----
12
67
23

1:Insert_front
2>Delete_front
3:Insert_rear
4>Delete_rear
5>Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
enter the choice
5
-----
Enter the position:
1

1:Insert_front
2>Delete_front
3:Insert_rear
4>Delete_rear
5>Delete at specified position
6:Insert at specified position
7:Display_list
8:EXIT
enter the choice
7
-----
67
23
```