

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 struct node
4 {
5     int info;
6     struct node *link;
7 };
8 typedef struct node *NODE;
9 NODE getnode()
10 {
11     NODE x;
12     x=(NODE)malloc(sizeof(struct node));
13     if(x==NULL)
14     {
15         printf("mem full\n");
16         exit(0);
17     }
18     return x;
19 }
20 int freenode(NODE x)
21 {
22     free(x);
23     return 0;
24 }
25 NODE insert_front(NODE first,int item)
26 {
27     NODE temp;
28     temp=getnode();
29     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 delete_front(NODE first)
38 {
39     NODE temp;
40     if(first==NULL)
41     {
42         printf("list is empty cannot delete\n");
43         return first;
44     }
45     temp=first;
46     temp=temp->link;
47     printf("item deleted at front-end is=%d\n",first->info);
48     free(first);
49     return temp;
50 }
51 NODE insert_rear(NODE first,int item)
52 {
53     NODE temp,cur;
54     temp=getnode();
55     temp->info=item;
56     temp->link=NULL;
57     if(first==NULL)
58         return temp;
```



```
59     cur=first;
60     while(cur->link!=NULL)
61     {
62         cur=cur->link;
63         cur->link=temp;
64     }
65     NODE delete_rear(NODE first)
66     {
67         NODE cur,prev;
68         if(first==NULL)
69         {
70             printf("list is empty cannot delete\n");
71             return first;
72         }
73         if(first->link==NULL)
74         {
75             printf("item deleted is %d\n",first->info);
76             free(first);
77             return NULL;
78         }
79         prev=NULL;
80         cur=first;
81         while(cur->link!=NULL)
82         {
83             prev=cur;
84             cur=cur->link;
85         }
86         printf("item deleted at rear-end is %d",cur->info);
87         free(cur);
```



main.c

```
87     tree(cur);
88     prev->link=NULL;
89
90     return first;
91 }
92 void display(NODE first)
93 {
94     NODE temp;
95     if(first==NULL)
96         printf("list empty cannot display items\n");
97     for(temp=first;temp!=NULL;temp=temp->link)
98     {
99         printf("%d\n",temp->info);
100    }
101 }
102 NODE delete_pos(int pos, NODE first)
103 {
104     NODE cur;
105     NODE prev;
106     int count;
107     if(first==NULL || pos<=0)
108     {
109         printf("invalid position \n");
110         return NULL;
111     }
112     if (pos==1)
113     {
114         cur=first;
115         first=first->link;
```



main.c

```
115     first=first->link;
116     freenode(cur);
117     return first;
118 }
119 prev=NULL;
120 cur=first;
121 count=1;
122 while(cur!=NULL)
123 {
124     if(count==pos) break;
125     prev=cur;
126     cur=cur->link;
127     count++;
128 }
129 if(count!=pos)
130 {
131     printf("invalid position \n");
132     return first;
133 }
134 if(count!=pos)
135 {
136     printf("invalid position specified \n");
137     return first;
138 }
139 prev->link=cur->link;
140 freenode(cur);
141 return first;
142 }
143 int main()
```



main.c

```
144  {
145  int item,choice,pos;
146  NODE first=NULL;
147  system("cls");
148  for(;;)
149  {
150  printf("\n 1:Insert_front\n 2:Delete_front\n"
151  "3:Insert_rear\n 4:Delete_rear\n 5.Delete at specified
position \n 6:Display_list\n 7:Exit\n");
152  printf("enter the choice\n");
153  scanf("%d",&choice);
154  switch(choice)
155  {
156  case 1:printf("enter the item at front-end\n");
157  scanf("%d",&item);
158  first=insert_front(first,item);
159  break;
160  case 2:first=delete_front(first);
161  break;
162  case 3:printf("enter the item at rear-end\n");
163  scanf("%d",&item);
164  first=insert_rear(first,item);
165  break;
166  case 4:first=delete_rear(first);
167  break;
168  case 5:printf("enter the position of the item to be
deleted: ");
169  scanf("%d",&pos);
170  first=delete_pos(pos,first);
```



main.c

```
• clang-7 -pthread -lm -o main main.c
• ./main
sh: 1: cls: not found
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5.Delete at specified position
6:Display_list
7:Exit
```

```
enter the choice
```

```
1
```

```
enter the item at front-end
```

```
34
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
• clang-7 -pthread -lm -o main main.c
• ./main
sh: 1: cls: not found
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5.Delete at specified position
6:Display_list
```

```
7:Exit
```

```
enter the choice
```

```
1
```

```
enter the item at front-end
```

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit
```

enter the choice

3

enter the item at rear-end

22

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7: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.Delete at specified position  
6:Display_list  
7:Exit
```

enter the choice

4

item deleted is 22

Console

Shell



```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit
```

enter the choice

5

enter the position of the item to be deleted:

1

invalid position

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit
```

enter the choice

6

list empty cannot display items

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit
```

enter the choice