

## PRACTICE PROG:

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4
5 struct node
6 {
7     int info;
8     struct node *link;
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("mem 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;
```

```
31 temp->link=NULL;
32 if(first==NULL)
33     return temp;
34 temp->link=first;
35 first=temp;
36 return first;
37 }
38
39 NODE delete_rear(NODE first)
40 {
41     NODE cur,prev;
42     if(first==NULL)
43     {
44         printf("list is empty cannot delete\n");
45         return first;
46     }
47     if(first->link==NULL)
48     {
49         printf("item deleted is %d\n",first->info);
50         free(first);
51         return NULL;
52     }
53     prev=NULL;
54     cur=first;
55     while(cur->link!=NULL)
56     {
57         prev=cur;
58         cur=cur->link;
59     }
60     printf("item deleted at rear-end is %d",cur->info);
61     free(cur);
```

```

60 printf("Item deleted at Rear-end is %d",cur->info);
61 free(cur);
62 prev->link=NULL;
63 return first;
64 }
65 NODE asc(NODE first)
66 {
67     NODE prev=first;
68     NODE cur=NULL;
69     int temp;
70
71     if(first== NULL) {
72         return 0;
73     }
74     else {
75         while(prev!= NULL) {
76
77             cur = prev->link;
78
79             while(cur!= NULL) {
80
81                 if(prev->info > cur->info) {
82                     temp = prev->info;
83                     prev->info = cur->info;
84                     cur->info = temp;
85                 }
86                 cur = cur->link;
87             }
88             prev= prev->link;
89         }
90     }

```

```

91         return first;
92     }
93     NODE des(NODE first)
94     {
95         NODE prev=first;
96         NODE cur=NULL;
97         int temp;
98
99         if(first==NULL) {
100             return 0;
101         }
102         else {
103             while(prev!= NULL) {
104
105                 cur = prev->link;
106
107                 while(cur!= NULL) {
108
109                     if(prev->info < cur->info) {
110                         temp = prev->info;
111                         prev->info = cur->info;
112                         cur->info = temp;
113                     }
114                     cur = cur->link;
115                 }
116                 prev= prev->link;
117             }
118             return first;
119         }
120     }
121 int count(NODE first)

```

```

120     }
121 int count(NODE first)
122 {
123     NODE cur;
124     cur=first;
125     int c;
126     if(first==NULL)
127     {
128         c=0;
129
130         return c;
131     }
132     if(first->link==NULL)
133     {
134         c=1;
135         return c;
136     }
137     while(cur!=NULL)
138     {
139
140         c++;
141         cur=cur->link;
142     }
143     return c;
144 }
145
146
147 void search(NODE first,int ele)
148 {
149     NODE cur;
150     cur=first;

```

```

150     cur=first;
151     int flag=0;
152     if(first==NULL)
153     {
154         printf("list empty can't search \n");
155         return ;
156     }
157     while(cur!=NULL)
158     {
159         if(ele==cur->info)
160         {
161             flag=1;
162             break;
163         }
164         cur=cur->link;
165     }
166     if(flag==1)
167         printf("SEARCH SUCCESSFULL \n");
168     else
169         printf("SEARCH UNSUCCESSFULL \n");
170
171 }
172
173 void display(NODE first)
174 {
175     NODE temp;
176     if(first==NULL)
177         printf("list empty cannot display items\n");
178     for(temp=first;temp!=NULL;temp=temp->link)
179     {
180         printf("%d\t",temp->info);

```

```

179     {
180         printf("%d\n",temp->info);
181     }
182 }
183 void main()
184 {
185     int item,choice,choice2,j,key;
186     NODE first=NULL;
187
188
189     for(;;)
190     {
191         printf("\n 1:Insert_front\n 2:Delete_rear\n 3:count \n 4:sort\n 5:search\n 6:display_list\n 7:Exit\n");
192         printf("enter the choice\n");
193         scanf("%d",&choice);
194         switch(choice)
195         {
196             case 1:
197                 printf("enter item to be inserted at front end \n");
198                 scanf("%d",&item);
199                 first=insert_front(first,item);
200                 break;
201             case 2:
202                 delete_rear(first);
203                 break;
204             case 3:
205                 j=count(first);
206                 printf("no of items in list: %d ",j);
207                 break;
208             case 4:
209                 printf("press 1 for ascending order and 2 for descending order \n");

```

```

208             case 4:
209                 printf("press 1 for ascending order and 2 for descending order \n");
210                 scanf("%d",&choice2);
211                 if(choice2==1)
212                     first=asc(first);
213                 if(choice2==2)
214                     first=des(first);
215                 break;
216             case 5:
217                 printf("enter element to be serached for \n");
218                 scanf("%d",&key);
219                 search(first,key);
220                 break;
221             case 6:
222                 display(first);
223                 break;
224             default:exit(0);
225                 break;
226         }
227     }
228     getch();
229 }
230
231

```

## OUTPUT:

```
1:Insert_front
2:Delete_rear
3:count
4:sort
5:search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
5

1:Insert_front
2:Delete_rear
3:count
4:sort
5:search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
4

1:Insert_front
2:Delete_rear
3:count
4:sort
```

```
4:sort
5:search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
3

1:Insert_front
2:Delete_rear
3:count
4:sort
5:search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
2

1:Insert_front
2:Delete_rear
3:count
4:sort
5:search
6:display_list
7:Exit
enter the choice
```

```
enter the choice
1
enter item to be inserted at front end
1

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
6
1
2
3
4
5

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
3
```

```
enter the choice
3
no of items in list: 5
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
4
press 1 for ascending order and 2 for descending order
2

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
6
5
4
3
2
1
```

```
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
5
enter element to be serached for
20
SEARCH UNSUCCESSFULL

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
5
enter element to be serached for
3
SEARCH SUCCESSFULL

1:Insert_front
2:Delete_rear
```

```
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
7

...Program finished with exit code 0
Press ENTER to exit console.
```