```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)
```

```
33 NODE insert_front(NODE first,int item)
34 {
35 NODE temp;
36 temp=getnode();
37 temp->info=item;
38 temp->link=NULL;
39 if(first==NULL)
40 return temp;
41 temp->link=first;
42 first=temp;
43 return first;
44 }
45
46 NODE delete_rear(NODE first)
47 {
48 NODE cur,prev;
49 if(first==NULL)
50 {
51 printf("List is empty. Cannot delete\n");
52 return first;
53 }
54 if(first->link==NULL)
55 {
56 printf("Item deleted is %d\n",first->info);
57 free(first);
58 return NULL;
59 }
60
```

```
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("Item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List is empty. \n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}

int length(NODE first)
{
```

```
{
    NODE cur;
    int count=0;
    if(first==NULL)
    return 0;
    cur=first;
    while(cur!=NULL)
    {
        count++;
        cur=cur->link;
    }
    return count;
}

void search(int key,NODE first)
{
    NODE cur;
    if(first==NULL)
    {
        printf("List is empty\n");
        return;
    }
    cur=first;
    while(cur!=NULL)
    {
        if(key==cur->info)
        break;
```

```
111         break;
112         cur=cur->link;
113     }
114     if(cur==NULL)
115     {
116         printf("Search is unsuccessful\n");
117         return;
118     }
119     printf("Search is successful. Element is found in the list at position %d \n",cur->link);
120 }
121
122 NODE asc(NODE first)
123 {
124     NODE prev=first;
125     NODE cur=NULL;
126             int temp;
127
128 if(first== NULL) {
129         return 0;
130         }
131 else {
132       while(prev!= NULL) {
133
134         cur = prev->link;
135
136         while(cur!= NULL) {
137
138
```

```
        if(prev->info > cur->info) {
       temp = prev->info;
       prev->info = cur->info;
       cur->info = temp;
        }
        cur = cur->link;
       }
       prev= prev->link;
                }
            }
            return first;
        }

        NODE des(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
            int temp;

if(first==NULL) {
    return 0;
     }
    else {
      while(prev!= NULL) {

      cur = prev->link;

```

```
164
165         while(cur!= NULL) {
166
167          if(prev->info < cur->info) {
168         temp = prev->info;
169         prev->info = cur->info;
170         cur->info = temp;
171             }
172             cur = cur->link;
173             }
174             prev= prev->link;
175              }
176              }
177              return first;
178          }
179
180  int main()
181  {
182  int item,choice,count,key,option;
183  NODE first=NULL;
184  for(;;)
185  {
186      printf("\n 1:Insert front\n 2:Delete rear\n 3:Display list\n 4:Count items\n 5:Search items\n 6:Order
187      printf("Enter the choice\n");
188      scanf("%d",&choice);
189      switch(choice)
190      {
191
```

```
189     switch(choice)
190     {
191         case 1:
192             printf("Enter the item at front-end\n");
193             scanf("%d",&item);
194             first=insert_front(first,item);
195             break;
196         case 2:
197             first=delete_rear(first);
198             break;
199         case 3:
200             display(first);
201             break;
202         case 4:
203             count=length(first);
204             printf("Length of items in the list is %d\n ",count);
205             break;
206         case 5:
207             printf("Enter the item to be searched\n");
208             scanf("%d",&key);
209             search(key,first);
210             break;
211         case 6:
212             printf("\n1:Ascending ordered list\n2:Descending ordered list\n");
213             scanf("%d",&option);
214             if(option==1)
215             {
```

```
            printf("Length of items in the list is %d\n ",count);
            break;
        case 5:
            printf("Enter the item to be searched\n");
            scanf("%d",&key);
            search(key,first);
            break;
        case 6:
            printf("\n1:Ascending ordered list\n2:Descending ordered list\n");
            scanf("%d",&option);
            if(option==1)
            {
                first=asc(first);
                display(first);
            }
            else
            {
                first=des(first);
                display(first);
            }
            break;
        default:
        exit(0);
    }
}
return 0;
}
```

```
main.c:119:77: warning: format '%d' expects argument of type 'int', but argument 2 has type 'struct node *' [-Wformat=
]

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
1
Enter the item at front-end
21

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
1
Enter the item at front-end
32
```

```
 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
1
Enter the item at front-end
22

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
1
Enter the item at front-end
54

 1:Insert front
 2:Delete rear
```

```
 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
6

1:Ascending ordered list
2:Descending ordered list
1
21
22
32
54

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
```

```
 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
6

1:Ascending ordered list
2:Descending ordered list
2
54
32
22
21

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
```

```
 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
3
54
22
32
21

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
4
Length of items in the list is 4

 1:Insert front
```

```
 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
5
Enter the item to be searched
55
Search is unsuccessful

 1:Insert front
 2:Delete rear
 3:Display list
 4:Count items
 5:Search items
 6:Order list
 7:Exit
Enter the choice
7


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