```
#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;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}


NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
```

```
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
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;
}


NODE order_list(int item,NODE first)
{
NODE temp,prev,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL) return temp;
if(item<first->info)
{
temp->link=first;
return temp;
}
prev=NULL;
cur=first;
while(cur!=NULL&&item>cur->info)
{
prev=cur;
cur=cur->link;
}
prev->link=temp;
temp->link=cur;
return first;
}


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 display(NODE first)
{
    printf("\n");

  NODE temp;
 if(first==NULL)
 printf("list empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}


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;
cur=cur->link;
}
if(cur==NULL)
{
printf("search is unsuccessful\n");
return;
}
printf("search successfull\n");
}


int main()
{
int item,choice,pos,count;
```

```
NODE first=NULL;
int n,i;
NODE a,b;


for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_rear\n");
printf(" 3:Display_list\n 4:Count items /n 5:Sort_list\n ");
printf("enter the choice\n");
scanf("%d",&choice);

switch(choice)
 {
  case 1:printf("enter the item at front-end\n");
       scanf("%d",&item);
       first=insert_front(first,item);
       break;

  case 2:first=delete_rear(first);
       break;

  case 3:display(first);
       break;

    case 4:count=length(first);
       printf("length(items) in the list is %d\n",count);
       break;

  case 5:printf("enter the item to be inserted in ordered_list\n");
       scanf("%d",&item);
       first=order_list(item,first);
       break;

  default:exit(0);
       break;
 }
}

}
```

```
 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
1
enter the item at front-end
15

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
1
enter the item at front-end
489

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
1
enter the item at front-end
1584

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
3

1584
489
15

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
2
```

```
enter the item to be inserted in ordered_list
15

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
5
enter the item to be inserted in ordered_list
84

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
5
enter the item to be inserted in ordered_list
-15

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
5
enter the item to be inserted in ordered_list
450

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
3

-15
15
84
450

 1:Insert_front
 2:Delete_rear
 3:Display_list
 4:Count items /n 5:Sort_list
 enter the choice
4
length(items) in the list is 4
```