```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;
}

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("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

NODE order_list(NODE first)
{
    int swapped, i;
    NODE ptr1,lptr=NULL;

      if (first == NULL)
      return first;

    do
    {
        swapped = 0;
        ptr1 = first;

        while (ptr1->link != lptr)
        {
            if (ptr1->info > ptr1->link->info)
            {
              int temp = ptr1->info;
              ptr1->info = ptr1->link->info;
              ptr1->link->info = temp;
              swapped = 1;
            }
            ptr1 = ptr1->link;
        }
        lptr = ptr1;
    }
    while (swapped);
    return first;
}
```

```
void count(NODE first){
    NODE temp;
    temp=first;
    int c=0;
    while(temp!=NULL){
    temp=temp->link;
    c++;
    }
    printf("Number of elements: %d\n",c);
}

void list_search(NODE first, int key){
    NODE temp;
    temp=first;
    int c=0,f=0;
    while(temp!=NULL){
        c++;
        if(temp->info==key){
            printf("Search successful, element position: %d\n",c);
            f=1;break;
        }
        temp=temp->link;
    }
    if(f==0)
    printf("Search Unsuccessful!\n");
}
void display(NODE first)
{
 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);
  }
}
int main(){
    int item,choice,pos,i,n;
    NODE first=NULL;
    for(;;)
        {
        printf("1.insert-
front\n2.delete_rear\n3.display\n4.count items\n5.search\n6.order\nAny other key to exit\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(first);
  break;
  case 5:printf("Enter element to be searched: ");
  scanf("%d",&item);
  list_search(first,item);
  break;
  case 6:
  first=order_list(first);
  break;
 default:exit(0);
 }
}
}
```

Output:

```
1
enter the item at front-end
10
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
1
enter the item at front-end
30
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
1
enter the item at front-end
40
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
3
40
30
10
20
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
```

```
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
4
Number of elements: 4
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
5
Enter element to be searched: 10000
Search Unsuccessful!
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
6
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
3
10
20
30
40
1.insert-front
2.delete_rear
```