```python
#선택정렬1
def selectionsort1(dat):
  size=len(datalist)
  cnt=0
  for i in range(0,size-1,1): #i=0~size-2
    for j in range(i+1,size,1): #j=i+1~size-1(1~4)
      if datalist[i]>datalist[j]: #교환 조건
        datalist[i],datalist[j]=datalist[j],datalist[i]
        cnt += 1 #데이터 교환 횟수 증가
    print(i+1,"단계:",datalist)
    print('-'*30)

  print("총 데이터 교환 횟수 :",cnt)

#메인코드
print("[선택정렬1 - 오름차순]")
datalist=[5,4,2,1,3]
print("\n선택정렬1 전:",datalist)
selectionsort1(datalist)
print("\n선택정렬1 후:",datalist)
```

```python
#선택정렬2
def selectionsort2(nlist):
  size=len(nlist)
  cnt=0 #데이터 교환 횟수 변수

  for i in range(size-1): #i=0~size-2
    mindex=i
    for j in range(i+1,size): #j=0~size-1
      if nlist[mindex]>nlist[j]: #교환 조건
        mindex=j #단계별 작은 값의 인덱스로 교환
    if mindex!=i:
      nlist[i],nlist[mindex]=nlist[mindex],nlist[i]
      cnt+=1 #데이터 교환 횟수 증가
    print(i+1,"단계:",nlist)
    print("-"*30)

  print("총 데이터 교환 횟수:",cnt)
```

```python
#메인코드
print("[선택정렬2 - 오름차순]")
datalist=[5,4,2,1,3]
print("선택정렬2 전:",datalist)
selectionsort2(datalist) #선택정렬2 함수 호출
print("선택정렬2 후:",datalist)
```

```python
#삽입정렬
def insertionsort(nlist):
  size=len(nlist)

  for i in range(1,size): #데이터 삽입 순서: 2번째데이터~마지막데이터
    indata=nlist[i] #삽입되는 데이터
    pos=i #삽입되는 데이터의 위치(index)

    while pos>0 and nlist[pos-1]>indata:
      nlist[pos]=nlist[pos-1]
      pos -= 1 #pos=pos-1

    if pos != i: #pos가 i가 아니라면 indata를 위치에 저장
      nlist[pos]=indata

    print(i,"단계:",nlist)
```

```python
    print("-"*30)

#메인코드
print("[삽입정렬 - 오름차순]")
datalist=[5,4,2,1,3]
print("\n삽입정렬 전:",datalist)
insertionsort(datalist) #삽입정렬 함수 호출
print("\n삽입정렬 후:",datalist)
```

```python
#함수 정의
def SearchList(k,alist):
    size=len(alist) #리스트의 길이

    for i in range(size): #range(0,size,1)
        if k==alist[i]:
            return i
    return -1

#메인코드
print('[순차 탐색]\n')
datalist = [5,4,2,1,3]
print("리스트 데이터 :",datalist)
key=int(input("탐색 키(search key) 입력 : "))

index=SearchList(key,datalist) #순차 탐색 함수 호출

if index==-1:
    print("=> 탐색 실패")
else:
    print("=> 탐색 성공!!!\n=>탐색 위치(index) =",index)
```

```python
#함수 정의
def SearchList(k,alist):
    size=len(alist) #리스트의 길이

    for i in range(size): #range(0,size,1)
        print("%2d단계. "%(i+1),end="")
        if k==alist[i]:
            print("%d와(과) 인덱스 %d번째의 값 %d 비교 => 일치"%(k,i,alist[i]))
            return i
        else:
            print("%d와(과) 인덱스 %d번째의 값 %d 비교 => 불일치"%(k,i,alist[i]))
    return -1
```

```python
#메인코드
print('[순차 탐색]\n')
datalist = [5,4,2,1,3,7,6]
print("리스트 데이터 :",datalist)
key=int(input("\n탐색 키(search key) 입력 : "))

index=SearchList(key,datalist) #순차 탐색 함수 호출
print()
if index==-1:
    print("=> 탐색 실패")
else:
    print("=> %d회에 탐색 성공!!!\n=> 탐색 위치(index) = %d"%(index+1,index))
```

```python
#함수 정의
def SearchList(k,alist): #순차 탐색 함수
    size=len(alist) #리스트의 길이

    for i in range(size): #range(0,size,1)
        if k==alist[i]:
            return i
    return -1

#메인코드
print('[과일 탐색 프로그램]')
fruits = ["사과", "참외","배","수박","토마토","딸기","포도","바나나","메론","귤"]

while True:
    key=input("\n찾을 과일 입력 : ")
    if key=="끝":
        print("프로그램을 종료합니다.")
        break
    index=SearchList(key,fruits)
    print()
    if index==-1:
        print(key+"은(는) 과일 리스트에 없습니다.")
    else:
        print(key+"의 저장 위치(index)는 "+str(index)+"입니다.")
```

```
pstr = "A STRING SEARCHING EXAMPLE CONSISTING OF STING"
pattern = "STING"

print("주어진 문자열 :",pstr)
print("찾고자 하는 패턴 :",pattern)
print()
ssize = len(pstr) #주어진 문자열의 길이
psize = len(pattern) #패턴 길이

for i in range(ssize): #문자열의 길이까지 반복이 될수 있도록
  if len(pstr[i:])<psize: #남은 문자열의 길이가 패턴 길이보다 작으면 프로그램 종료
    print("Finish!!!")
    break

  cnt=0 #문자열과 패턴의 일치하는 개수 카운트 변수

  for j in range(psize): #패턴 길이까지 반복
    print("i=",i,"j=",j,"i+j=",i+j, pstr[i+j],pattern[j],"cnt=",cnt) #문자열 패턴 매칭 과정
    if pstr[i+j]==pattern[j]:
      cnt+=1 #일치되는 문자열 개수 카운트
  if cnt==psize: #일치하는 개수와 패턴 길이가 같으면
    print("Matched index",i)  #매칭
```

```
pstr = "A STRING SEARCHING EXAMPLE CONSISTING OF STING"
pattern = "STING"

print("주어진 문자열 :",pstr)
print("찾고자 하는 패턴 :",pattern)
print()
ssize = len(pstr) #주어진 문자열의 길이
psize = len(pattern) #패턴 길이
cnt=0 #여러개의 패턴에 대한 개수 카운트 변수

```

```
i=0 #초기값
while i<ssize: #문자열의 길이까지 반복
  temp=pstr[i:]
  if pattern in temp: #문자열 패턴이 포함된다면
    tempindex=temp.index(pattern) #인덱스 반환

    print("Matched index: ", i+tempindex)
    cnt+=1
    i=i+tempindex+1
  else:
    break
if cnt==0:
  print("No index Matched")
print("Finish!!!")
```

```python
def BubbleSort(blist):
  size=len(blist) #리스트의 길이
  cnt=0 #데이터 교환 횟수 카운트
  for i in range(0,size-1): #단계(회전), i=0,1,2,3 / 0~size-2
    for j in range(0,size-i-1): #데이터 교환에 대한 변수
      #i=0,j=0~3(=5-0-2)/ i=1,j=0~2(=5-1-2) /i=2,j=0~1 / i=3,j=0
      if blist[j]>blist[j+1]: #데이터 교환 조건
        temp=blist[j]
        blist[j]=blist[j+1]
        blist[j+1]=temp
        #blist[j],blist[j+1]=blist[j+1],blist[j]
        cnt+=1 #데이터 교환 횟수 증가
    print(i+1,"단계:",blist)
    print("-"*30)
  print("총 데이터 교환 횟수 :",cnt)

#메인코드
datalist=[5,4,2,1,3]
print("[버블정렬 - 오름차순]")
print("\n버블정렬 전 :",datalist)
print()
BubbleSort(datalist)
print("\n버블정렬 후 :",datalist)
```

```python
menu=["탕수육","유린기","팔보채","유산슬","라조기","고추잡채","깐풍기","깐쇼새우"] #8개 메뉴

for i in range(0,len(menu)):
  for j in range(i+1,len(menu)):
    for k in range(j+1,len(menu)):
      print(menu[i],menu[j],menu[k])
```

```python
#이진 탐색 함수 정의
def BinarySearchASC(sk, slist):#오름차순으로 정렬된 데이터를 이진 탐색 수행
  low=0 ; high=len(sllist)-1

  while low<=high:
    mid=int((low+high)/2) #중간값(index) 계산

    if sk<slist[mid]:
      high=mid-1
    elif sk>sllist[mid]:
      low=mid+1
    else:  #sk==slist[mid]
      print("=>일치")
      return mid

  return -1 #탐색 실패
```

```python
print('[이진 탐색(오름차순정렬)]\n')
numlist=[43,97,13,96,84,51,64,25,72,14,93,33,6,95,53]
print("원본 데이터:",numlist)
numlist.sort() #오름차순 정렬
print("정렬 데이터:",numlist)
cnt=0 #탐색 횟수 카운트 변수
key=int(input("\n탐색 키(search key) 입력:"))
result = BinarySearchASC(key, numlist)#이진 탐색 함수 호출
print()

if result==-1:
  print("=>탐색 실패")
else:
  print("=>%d번에 탐색 성공!!!"%cnt)
  print("=>탐색 위치(index)=%d"%result)
```

```
#이진 탐색 함수 정의
def BSearchASC(sk,slist,low,high): #재귀함수
  global cnt
  if low>high:
    return -1  #재귀함수 호출 종료

  mid=int((low+high)/2) #중간인덱스 계산
  cnt+=1

  if sk==slist[mid]:
    print("key=%d,slist[%d]=%d => 일치"%(sk,mid,slist[mid]))
    return mid
  elif sk>slist[mid]:
    low=mid+1
    print("key=%d,slist[%d]=%d => 불일치"%(sk,mid,slist[mid]))
  else:
    high=mid-1
    print("key=%d,slist[%d]=%d => 불일치"%(sk,mid,slist[mid]))

  return BSearchASC(sk,slist,low,high)
```

```
print('[이진 탐색(오름차순정렬)]\n')
numlist=[43,97,13,96,84,51,64,25,72,14,93,33,6,95,53]
print("원본 데이터:",numlist)
numlist.sort() #오름차순 정렬
print("정렬 데이터:",numlist)
cnt=0 #탐색 횟수 카운트 변수
start=0 #시작 인덱스
end=len(numlist)-1 #마지막 인덱스
key=int(input("\n탐색 키(search key) 입력:"))
result = BSearchASC(key,numlist,start,end)#이진 탐색 함수 호출
print()
```

```
if result==-1:
  print("=>탐색 실패")
else:
  print("=>%d번에 탐색 성공!!!"%cnt)
  print("=>탐색 위치(index)=%d"%result)
```

#함수 작성

```
def BinarySearchDESC(sk, slist):#오름차순으로 정렬된 데이터를 이진 탐색 수행
  global cnt
  low=0 ; high=len(slist)-1

  while low<=high:
    mid=int((low+high)/2) #중간값(index) 계산
    cnt+=1 #탐색 횟수 증가
    if sk>slist[mid]:
      high=mid-1
    elif sk<slist[mid]:
      low=mid+1
    else:  #sk==slist[mid]
      return mid

  return -1 #탐색 실패

print('[과일 찾기 프로그램]')
flist= ["사과", "참외","배","수박","토마토","딸기","포도","바나나","메론","귤"]
flist.sort(reverse=True) #내림차순 정렬 - 이진 탐색의 선행작업
```

```
print('[과일 찾기 프로그램]')
flist= ["사과", "참외","배","수박","토마토","딸기","포도","바나나","메론","귤"]
flist.sort(reverse=True) #내림차순 정렬 - 이진 탐색의 선행작업

while True:
  key=input("찾을 과일 입력 : ")
  cnt=0 #탐색 횟수 초기화

  if key=='끝':
    print("프로그램을 종료합니다.")
    break

  result=BinarySearchDESC(key,flist) #함수 호출

  if result==-1:
    print(key+"은(는) 과일 리스트에 없습니다.")
  else:
    print(key+"를 "+str(cnt)+'회 탐색으로 찾았습니다!!!')
    print(key+'의 저장 위치(index)는 '+str(result)+'입니다.')
```