1.

```
import operator

inp = int(input())
lis=[]
for x in range(inp):
    a, b= map(int,input().split())
    lis.append([a,b])

lis.sort(key = operator.itemgetter(1)) // y값 정렬 후
lis.sort(key = operator.itemgetter(0)) // x 값 정렬

for adf in lis:
    print(adf[0],adf[1])
```

map(function_to_apply, list_of_inputs) : 입력받은 자료형의 각 요소가 함수(function_to_apply)에 의해 수행된 결과를 묶어서 map iterator 객체로 리턴

list.sort() : list 내부에서 정렬, 리스트 형에 한해서만 동작

list.sort(reverse = True) : 내림차순 정렬

list.sort(key = len) : key 옵션의 지정된 함수의 결과에 따라 정렬, 원소의 길이에 따라 정렬

operator.itemgetter(x) : 인덱스 x의 값을 가져옴

[y값 정렬 후 x 값 정렬 하는 이유]

입력이

2

9 3

11 1 일 때

x 값부터 정렬할 경우 (11, 1) (9, 3) 순으로 출력되어진다. 따라서 이를 방지하기위해 y값부터 정렬해준 후, x값을 정렬해준다.

2.

```
dic = {}
N, M = map(int, input().split())
for i in range (N):
    st = input()
    dic[st] = i+1
listname =  list(dic.keys())
for j in range(M):
    inp = input()
    if inp not in dic:
        print(listname[int(inp)-1])
    else:
        print(dic[inp])

N = int(input())
print(4*N)
```

딕셔너리

기본구조

{key1:value1, key2:value2, key3:value3 .....}

key에는 변하지 않는 값을 사용하고, value에는 변하는 값과 변하지 않는 값 모두 사용할 수 있다.

딕셔너리에서 값을 참조할때는 리스트처럼 인덱스 값을 이용하는 것이 아닌 **key값**을 이용하여 value값을 가져온다.

딕셔너리는 sequential 구조가 아니다.

dic[key] = value -> 딕셔너리 안에 key:value 쌍이 추가된다.

del dic[key] : 딕셔너리 안에 있는 key:value 쌍이 삭제된다.

dic[key] : 어떤 key의 value를 얻기 위해 사용

key는 고유한 값이므로 중복되는 key값을 설정해 놓으면 하나를 제외한 나머지 것들이 모두 무시된다.

dic.keys() : 모든 key값을 리스트로 반환해준다

dic.values() : 모든 value값을 리스트로 반환해준다.

dic.items() : 모든 key와 value의 쌍을 튜플로 묶은 값을 리스트로 반환해준다.

3.

```
string = str(input())
wordlen = 0
res = 0
for x in string:
    if(x == 'U' and wordlen == 0):
        res += 1
        wordlen += 1
    elif(x == 'C' and (wordlen == 1 or wordlen == 3)): //C는 두번째와
네번째에만 들어간다.
        res += 1
        wordlen += 1
    elif(x == 'P' and wordlen == 2):
        res += 1
        wordlen += 1

if(res == 4):
    print("I love UCPC")
else:
    print("I hate UCPC")
```