문제. 세금

지난 해 한 국가에서는 N명이 세금을 납부했다. 세금을 관리하는 직원은 지난 해 사람들이 세금을 낸 정보를 수합하는 과정에서 K번의 **정보 수집**을 진행하고자 한다. 각 정보 수집에서는 X와 Y 두 가지 변수가 사용되는데, 이는 납부한 세금의 금액이 X 이상 Y 이하인 사람의 수를 계산하는 작업을 의미한다.

각 K번의 정보 수집에 대하여, 납부한 세금의 금액이 X 이상 Y 이하인 사람의 수를 차례대로 계산하는 프로그램을 작성하여라.

예를 들어 $N = 5$명이 납부한 세금의 금액이 다음의 표와 같다고 해보자.

번호	1	2	3	4	5
금액	7	2	3	5	2

이후에 $K = 3$번의 정보 수집에 대한 정보가 다음과 같다고 가정하자. 정보 수집을 위해 사용되는 변수는 $[X, Y]$ 배열 형태로 주어진다.

번호	1	2	3
수집 내용	[1, 100]	[3, 5]	[2, 2]

현재 예시에서 각 정보 수집에 대한 정답 결과는 다음과 같다. 현재 예시에서 정보 수집 결과는 차례대로 5명, 2명, 2명이다.

번호	1	2	3
수집 결과	5	2	2

입력 조건

가장 먼저 세금을 낸 사람의 수 N과 정보 수집 횟수 K가 주어진다. 이때 N과 K는 모두 4 이상 100,000 이하의 자연수다.

이어서 N명의 사람들이 낸 세금에 대한 정보가 담긴 배열 arr가 주어진다. 이때, 각 사람이 납부한 세금의 금액은 1 이상 1,000,000,000 이하의 자연수다.
이어서 K회의 정보 수집 작업에 대한 정보가 담긴 배열 $queries$가 주어진다. 구체적으로 행의 크기가 N이고, 열의 크기가 2인 형태의 2차원 배열이다. 각 정보 수집 작업에서의 X와 Y는 모두 1 이상 1,000,000,000 이하의 자연수로, 이때 X는 Y보다 작거나 같다.

출력 조건

K번의 정보 수집에 대하여, 납부한 세금의 금액이 X 이상 Y 이하인 사람의 수를 차례대로 계산하여 1차원 배열 형태로 반환한다.

입출력 예시

N	K	arr	$queries$	정답
5	3	[7, 2, 3, 5, 2]	[[1, 100], [3, 5], [2, 2]]	[5, 2, 2]
6	4	[9, 8, 1, 5, 5, 5]	[[5, 8], [3, 10], [1, 2], [5, 9]]	[4, 5, 1, 5]

해설 2. 세금

본 문제는 전형적인 **정렬 및 이진 탐색 유형**의 문제다. 최대 100,000개의 정보 수집 작업이 수행될 수 있다는 점에서 선형 탐색으로는 시간 제한을 통과할 수 없다. 따라서 이진 탐색을 수행하기 위한 사전 조건으로, 먼저 N명이 납부한 세금 값에 대하여 오름차순 정렬을 수행한다. 이후에 매 정보 수집 작업에 대하여 이진 탐색을 활용하면 값의 범위가 [X, Y]에 속하는 원소의 개수를 구하기 위해 **로그(log) 시간이 소요**된다. 결과적으로 K번의 작업에 대하여 매번 이진 탐색을 수행하여 문제를 해결할 수 있다. 따라서 정렬을 위해 시간 복잡도 $O(N \cdot logN)$가 요구되며, 전체 쿼리에 대하여 시간 복잡도 $O(K \cdot logN)$가 요구되므로, 최종적인 시간 복잡도는 $O(NlogN + KlogN)$이다.

- **Python3 정답 코드 예시**

```
from bisect import bisect_left, bisect_right
```

```
# 값이 [left_value, right_value]인 데이터의 개수를 반환하는 함수
def count_by_range(array, left_value, right_value):
    right_index = bisect_right(array, right_value)
    left_index = bisect_left(array, left_value)
    return right_index - left_index

# 사람의 수(N), 정보 수집 횟수(K), 세금 정보 배열(arr), 쿼리 배열(queries)
def solution(N, K, arr, queries):
    arr.sort()  # 이진 탐색을 수행하기 위한 정렬
    answer = []
    for query in queries:
        x, y = query
        # 값이 [x, y] 범위에 있는 데이터의 개수 계산
        cnt = count_by_range(arr, x, y)
        answer.append(cnt)  # 결과 기록
    return answer
```

- **Java 정답 코드 예시**

```
import java.util.*;

class Solution {
    public static int lowerBound(int[] arr, int target, int start, int end) {
        while (start < end) {
            int mid = (start + end) / 2;
            if (arr[mid] >= target) end = mid;
            else start = mid + 1;
        }
        return end;
    }

    public static int upperBound(int[] arr, int target, int start, int end) {
        while (start < end) {
            int mid = (start + end) / 2;
            if (arr[mid] > target) end = mid;
            else start = mid + 1;
        }
        return end;
    }
```

```
    // 값이 [left_value, right_value]인 데이터의 개수를 반환하는 함수
    public static int countByRange(int[] arr, int leftValue, int rightValue) {
        // 유의: lowerBound와 upperBound는 end 변수의 값을 배열의 길이로 설정
        int rightIndex = upperBound(arr, rightValue, 0, arr.length);
        int leftIndex = lowerBound(arr, leftValue, 0, arr.length);
        return rightIndex - leftIndex;
    }

    // 사람의 수(N), 정보 수집 횟수(K), 세금 정보 배열(arr), 쿼리 배열(queries)
    public static int[] solution(int N, int K, int[] arr, int[][] queries) {
        Arrays.sort(arr); // 이진 탐색을 수행하기 위한 정렬
        int[] answer = new int[K];
        for (int i = 0; i < K; i++) {
            int x = queries[i][0];
            int y = queries[i][1];
            // 값이 [x, y] 범위에 있는 데이터의 개수 계산
            int cnt = countByRange(arr, x, y);
            answer[i] = cnt; // 결과 기록
        }
        return answer;
    }
}
```