Ranking

Rar the Cat has somehow managed to obtain a class list of **N** CS2040 students that contains the name of the students, as well as their scores for Sit In Lab 1.

Inspired by a NUSWhispers post that claims that teaching staff like to sort students by their marks then assign them letter grades, Rar the Cat decided to do so as well. However, he is too lazy to even sort it himself. He intends to hire you, a *supposedly* promising CS2040 student to generate the ***rank*** of each student in the class list.

> Aikenot Dueet, a distraught 2040 student whom we found crying in a cubicle, told us how sorting has effectively screwed his life. "At first, they sorted us by marks and assigned us letters based on a curve, and now, they are teaching us sorting while sorting us at the same time," weeped Aikenot.

Figure 1: Excerpt from the above-mentioned NUSWhispers post (Source: https://www.nuswhispers.com/confession/57486)

Let **S** be the number of students in the class list that have obtained a higher score than student **X**. The ***rank*** of student **X** in the class list is formally defined as **(S+1)**. This means that if there are many students that obtain the same score, they will all have the same rank.

Input
The first line of input contains a single integer **N**, the number of students in the class list.
N lines will follow. Each line will describe one student in the following form: `[name] [score]`.

Output
For each student, print the rank of the student in the following form: `[name] [rank]`.
These students should be printed in the **same order as the input**.

Limits

- $1 \leq \mathbf{N} \leq 50000$
- All the names of students will only contain uppercase and lowercase English letters with no spaces. The names will not be more than 20 characters long. It is possible that there can be 2 students with the same name.
- All scores of students will range from 0 to 10^9 inclusive.

Sample Input (**ranking1.in**)	Sample Output (**ranking1.out**)
4 Rar 200 Panda 200 Zip 100 Shark 300	Rar 2 Panda 2 Zip 4 Shark 1
Sample Input (**ranking2.in**)	Sample Output (**ranking2.out**)
4 SmartStudent 300 DumbStudent 300 BellCurveGod 300 SocCat 300	SmartStudent 1 DumbStudent 1 BellCurveGod 1 SocCat 1

Notes:

1. You should develop your program in the subdirectory **ex2** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – Ranking.java

You are given the below skeleton file `Ranking.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
 * Name         :
 * Matric. No   :
 * PLab Acct.   :
 */

import java.util.*;

public class Ranking {
    private void run() {
        //implement your "main" method here
    }

    public static void main(String[] args) {
        Ranking newRanking = new Ranking();
        newRanking.run();
    }
}
```