Problem 4 – Cubic Assault

After countless of hours of preparation – artillery storage, quantum research, and planning trough encoded messages, time has finally come for a war with the Spherical Race. Cubic is on the front lines devastating the enemy forces. Someone, however, must give statistics to Cubic about the count of enemies on each front. You are best for this job.

You will be given as input lines containing, a region, the type of the soldiers at that region and their amount. You must **store statistics** about the **amount of meteors** Cubic needs to defeat in **every region**. Note that if at one region 1 000 000 Green Meteors gather, they **combine** into 1 Red Meteor. By the same logic, 1 000 000 Red Meteors get **combined** into 1 Black Meteor. Note also, that you might receive **several input lines** with information about **1 region**. In that case just **update that region's statistics**. Multiple values to one type **increase** that type's value each time.

The input data will come in the following format **{regionName} -> {meteorType} -> {count}.**

When you receive the command "**Count em all**", you must **end** the input sequence. You must print all the regions ordered by the **amount of Black Meteors** – descending, then by the **length of their names** – ascending, and lastly **alphabetically**. For every region you must print how many green, red and black meteors there. Order the printing of the types by **amount of defeated units** in them – descending, and if two are with the same value, order them **alphabetically**.

Input

- As input you will receive random amount of input lines containing information about Cubic's statistics.
- The input ends when you receive the command "Count em all".

Output

- The output is simple. You must print each region and the statistics about the 3 types of meteors in it.
- The format of output is :

  {regionName}

  -> {firstType} : {firstTypeCount}

  -> {secondType} : {secondTypeCount}

  -> {thirdType} : {thirdTypeCount}

- The order of each type depends on its count as specified above. All data must be ordered correctly.

Constraints

- The input numbers will be valid integers in the range $[-2^{31} + 1, 2^{31} - 1]$.
- The input will always be in the format specified above. There is no need to check it explicitly.
- Allowed time/memory: 100ms/16MB

Examples

Input	Output
Cubica -> Black -> 1 Cubica -> Red -> 1000 Spherica -> Green -> 1000000 Count em all	Cubica -> Red : 1000 -> Black : 1 -> Green : 0 Spherica -> Red : 1 -> Black : 0 -> Green : 0

Input	Output
Triangula Canyon -> Black -> 100 Diagonalica -> Red -> 999999 Ellipsetica -> Red -> 100000000 Diagonalica -> Black -> 99 Diagonalica -> Green -> 1000000 Count em all	Diagonalica -> Black : 100 -> Green : 0 -> Red : 0 Ellipsetica -> Black : 100 -> Green : 0 -> Red : 0 Triangula Canyon -> Black : 100 -> Green : 0 -> Red : 0