# Problem 4. Trainlands

<https://judge.softuni.bg/Contests/Practice/Index/742#1>

So a train goes from A to B… Okay that’s too mainstream. Trainlands! Trainlands is a game of trains, in which trains develop themselves into gigantic trains for the competition. It got too trainy so let’s get to the main logic.

The **train** has a **name** (**string**), and **wagons**. The **wagons** have a **name** (**string**) and **power** (**integer**).   
The input comes in one of the following formats:

{trainName} -> {wagonName} : {wagonPower}

**Creates** a **train** with the **given name** and **adds** a **wagon** to it with the **given name** and **power**. If the train **already exists**, it just **adds** the **given wagon** to it.

{trainName} -> {otherTrainName}

**Adds all** of the **wagons** from the **other train** to the **first train**, and **REMOVES** the **other train**. If the first train **does NOT exist**, create it, and then **add** the **wagons**. The other train will **ALWAYS** be **EXISTENT**.

{trainName} = {otherTrainName}

**Copies** the **other train’s wagons**, without affecting the other train. **Copying** means, that the **first train’s wagons** **become** the **same** as the **other train’s wagons**. If the first train **does NOT exist**, create it. The other train will **ALWAYS** be **EXISTENT**.

When you get the command “It’s Training Men!” you should end the input sequence and print all of the trains and their wagons. The **trains** must be **ordered** by **total wagon power**, in **descending order**, and by **wagon count** in **ascending order** as **secondary criteria**. For each train you should print its wagons, **ordered** by **wagon power** in **descending order**.

### Input

* The input comes in the form of commands in one of the formats specified above.
* The input ends when the command “It’s Training Men!” is entered.

### Output

* As output you must print all of the trains and their wagons ordered as specified above.
* The format of printing is:

Train: {trainName}

###{wagon1Name} – {wagon1Power}

###{wagon2Name} – {wagon2Power}

. . .

### Constrains

* The **names** of the **trains** and the **wagons** will be **strings**.
* The **names** may contain **any ASCII character**, except “ ”, “-”, “**:**”, “>”, “=”.
* The **wagon power** will be a valid **integer** in **range [0, 1.000.000]**.
* There will be **NO invalid input**.
* The **wagons** will always have **unique** **names** in the **scope** of their **train**.
* Allowed time / memory: 100ms / 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Kivil -> KAKA : 1387  Zone -> Gh : 4081  Kivil -> RAMZES666 : 4677  Desolator -> MiraclE~ : 8432  Zone -> Kivil  It's Training Men! | Train: Zone  ###RAMZES666 - 4677  ###Gh - 4081  ###KAKA - 1387  Train: Desolator  ###MiraclE~ - 8432 |
| Kepler -> MinD\_ContRoL : 3782  Daun -> Fn : 6816  Miner -> Gh : 1198  Miner -> Sccc : 9030  Miner -> KAKA : 7409  Anna -> Miner  Daun = Anna  It's Training Men! | Train: Daun  ###Sccc - 9030  ###KAKA - 7409  ###Gh - 1198  Train: Anna  ###Sccc - 9030  ###KAKA - 7409  ###Gh - 1198  Train: Kepler  ###MinD\_ContRoL - 3782 |