**Battle Manager -Dictionary**

Create a program that manages battles. You need to keep information about **people**, the **health** and **energy** they have. You will be receiving **lines** with commands until you receive the **"****Results"** command. There are three **possible** commands:

* **"Add:{personName}:{health}:{energy}":**
  + **Add** the **person**, his/her **health** and **energy** to your **records**. **If** person with the given name already **exists**, just increase the **health** of the **person** with the **current** one that is **given**.
* **"Attack:{attackerName}:{defenderName}:{damage}":**
  + **Check** if both people **exist** and if **they do**, **reduce** the **defender’s** **health** with the **damage** **given**. If the defender’s health reaches **0** or **less**, the **person** is **disqualified,** and you need to **remove** him/her from your **records** and **print** the following **message**:
    - **"{defenderName} was disqualified!"**
  + You also have to **reduce** the **attacker’s** **energy** **by** **1**. If it reaches **0**, he/she is **disqualified,** and you need to **remove** him/her from your **records** and **print** the following **message**:
    - **"****{attackerName} was disqualified!"**
* **"Delete:{username}":**
  + Delete **all** records of the **given user**, **if** he **exists**. If "**All**" is **given as username** - delete **all records** you have.

In the end, you have to **print the count of people left, each person** with his/her **health** and **energy** sorted in **descending order** by the **health** and **then by** their **name** in **ascending** order in the following format:

**People count: {count}**

**{personName} - {health} - {energy}**

**{personName} - {health} - {energy}**

## Input

* You will be receiving linesuntil you receive the **"Results"** command.
* The **health** is an **integer** number in the range [1...100000].
* The **energy** is an **integer** number in the range [1...100].
* The input will **always** be **valid**.

## Output

* Print the appropriate message after the **"Attack"** command, **if** someone is **disqualified**.
* Print the people with their **health and energy** in the **format** described above.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Add:Mark:1000:5 Add:Clark:1000:3 Attack:Clark:Mark:500 Add:Allison:2500:5 Attack:Clark:Mark:300 Add:Charlie:4000:10 Attack:Clark:Mark:500 Results | Mark was disqualified!  Clark was disqualified!  People count: 2  Charlie - 4000 - 10  Allison - 2500 - 5 |
| **Comments** | |
| First, we receive the "**Add:Mark:1000:5**" command, so we **add** the person **Mark** in our **records** with his **health** (1000) and **energy** (5). Then we do the **same** for **Clark**. Afterwards **Clark attacks Mark multiple times**, and **Mark’s health** is **below 0** and **Clark’s energy** drops to **0**, meaning **both** of them are **disqualified**. Meanwhile, **Allison** and **Charlie** were **added**. In the end, we **print** the results- **2** people, then the collection **ordered** as described above. | |
|  | |
| Add:Bonnie:3000:5  Add:Kent:10000:10 Add:Johny:4000:10 Attack:Johny:Bonnie:400 Add:Chicken:1000:1 Add:Rabbit:3000:5 Add:Buggy:1259:10  Delete:Kent Attack:Chicken:Rabbit:1000 Results | Chicken was disqualified!  People count: 4  Johny - 4000 - 9  Bonnie - 2600 - 5  Rabbit - 2000 - 5  Buggy - 1259 - 10 |
|  | |
| Add:Bonnie:3000:5 Add:Johny:4000:10 Delete:All  Add:Bonnie:3333:3 Results | People count: 1  Bonnie - 3333 - 3 |