# Problem 4 – Airport Statistics

*You started to work in company that outsources software. Recently the company was contracted to create a software for the new airport that is going be built in your town. Since you have just started, you received a simple task: create a program that generates statistics of the arrival/departure flights.*

Write a JS program that aggregates information about airplane traffic to and from an airport. You will receive an array of strings in the following format:

**`{planeID} {town} {passengersCount} {action}`**

- **`planeID`** – the ID of the plane
- **`town`** – name of the town, which the plane is **coming from**, or the town the plane is **going to**
- **`passerngersCount`** – the number of passengers on the plane (this element must be parsed as Number)
- **`action`** – could be either "**`land`**" or "**`depart`**"

Store in your program the IDs of planes that **land** and remove them when they **depart**. If a plane is **landing** (indicated by its **action**), then the passengers on it are **arrivals** to our airport. If it's **departing**, then the passengers are **departures**. ***Some of the records will not be valid!*** To be considered valid, a record must meet the following criteria:

- A plane can depart only if it **landed first**.
- A plane **cannot** land again if it's **already landed** – it must depart **between** landings.
- The same plane can land and depart **multiple times** (see the examples).

If any of the above rules is not followed **ignore** that input string. Your program should generate a report containing:

- The IDs of planes that remain at our airport (their last valid action is **land**). **Sort** them **alphabetically**.
- A list of towns with the number of **arrivals** and **departures** for each town, and all **unique** IDs of the planes that made the flights. **Sort** the towns by the number of **arrivals** (descending). If two towns have the same number, sort them **alphabetically** by **name** (ascending). **Sort** the list of planes for each town **alphabetically**.

See the examples for formatting details. Note the **default** behavior of **`Array.sort()`** in JavaScript will sort by ASCII code, which is **not the same** as alphabetical.

## Input

Data is passed to your program as an **array of strings** in the format described above.

## Output

Print on the **console** the aggregated data, **sorted** as described above, **formatted** as seen in the examples.

## Constraints

- The input will always be in the specified formats. There is no need to check it explicitly.
- Plane IDs can consist of **any** alphanumeric character.

## Examples

| Input | Output | Explanation |
|---|---|---|
| [<br>"Boeing474 Madrid 300 land",<br>"AirForceOne WashingtonDC 178 land",<br>"Airbus London 265 depart",<br>"ATR72 WashingtonDC 272 land",<br>"ATR72 Madrid 135 depart"<br>] | Planes left:<br>- AirForceOne<br>- Boeing474<br>WashingtonDC<br>Arrivals: 450<br>Departures: 0<br>Planes:<br>-- AirForceOne<br>-- ATR72<br>Madrid<br>Arrivals: 300<br>Departures: 135<br>Planes:<br>-- ATR72<br>-- Boeing474 | **Boing474 landed** with 300 passengers from Madrid. Later on, **ART72 departed**, flying to Madrid with 135 passengers.<br>From WashingtonDC there were two planes that **landed** with **total** of 450 passengers.<br>Airbus **cannot** depart, since it did not land beforehand. |

| Input | Output | Explanation |
|---|---|---|
| [<br>"Airbus Paris 356 land",<br>"Airbus London 321 land",<br>"Airbus Paris 213 depart",<br>"Airbus Ljubljana 250 land"<br>] | Planes left:<br>- Airbus<br>Paris<br>Arrivals: 356<br>Departures: 213<br>Planes:<br>-- Airbus<br>Ljubljana<br>Arrivals: 250<br>Departures: 0<br>Planes:<br>-- Airbus | **Airbus** landed two consecutive times – the second one is **ignored**. After it's **departure** it can land **again**. |