Problem 1 – Cubic Artillery

Cubic has taken charge of creating places where the, the weapons, armor and basically all the resources for the Cubic army, can be stored. He is creating massive bunkers which are capable of storing weapons. However every weapon has a different capacity, and the bunkers can only take up to a specified total capacity. That is why you have taken the task of supervising and managing the process.

You will be given **n** – an integer specifying the **bunkers' maximum capacity**. Then you will be given input lines which will contain **English alphabet letters** and **numbers**, separated by a **single space**. The **letters** represent the **bunkers** and the **numbers** – the **weapons** and their **capacity**. Weapons must be **stored** in the bunkers. The **first entered** bunker is the **first in which weapons are getting stored**. Every weapon takes **specific capacity**, equal to its number.

When a bunker **overflows** (it **cannot contain** a given weapon due to lack of enough **free capacity)**, it passes the weapon to the **next entered bunker**. If the **next bunker cannot contain** the weapon, it passes it to the next-next, and so on till the **last entered bunker**. If there is **no bunker** that **can hold** the given weapon, **ignore** that weapon.

If there are **no other bunkers** besides the current one, you must check if current weapon can actually be contained (its **needed capacity** is **less than** the **maximum capacity specified for the bunkers**). If the weapon **can be contained**, you must **make enough free capacity** to hold that weapon, **if there isn't already**. That is done by **removing weapons**, starting from **the first entered**, till the last. If the weapon **cannot be** contained, **ignore** the weapon.

If a bunker overflows you must **remove it**, and print it on the console, along with all of the weapons it **currently contains**. If there are no weapons, just print "**Empty**". After you've removed that bunker, **the next one** becomes the **first in the order** – weapons will **first be passed to it**. If there are **no other bunkers**, you must **NOT** remove the one that overflowed.

The input sequence **ends** when you receive the command "**Bunker Revision**".

Input

- The input will come in lines of letters and digits separated by a space.
- The input ends when you receive the command "**Bunker Revision**".

Output

- For output, you must print a bunker, every time it overflows, after removing it.
- The format is the following: **{bunker} -> {weapon1}, {weapossn2}…**
- Where {bunker} is the letter that corresponds to that bunker, and the weapons are the numbers.
- In case a bunker has no weapons, just print "**Empty**".

Constraints

- The bunkers will only be English alphabet letters.
- Each bunker's letter will always be unique.
- The integer n will be In the range [0, 500].
- The weapons will always be valid integers in the range [0, 500].
- Allowed time/memory: 100ms/16MB.

Examples

Input	Output

`60`
`a 20 20 b 20 1`
`Bunker Revision`
	`a -> 20, 20, 20`

Comment
“a” is the first entered bunker. Then we receive the weapons **20** and **20** which are passed to **“a”**. Then we get the bunker **“b”**. Then again we receive a weapon **20**. **“a”** still has enough capacity to hold the weapon so we store it there. Then we get the weapon **1**. **“a”** has capacity **60/60** – it overflows, so we pass the weapon to the next bunker. We find **“b”** and we pass the weapon to **“b”**. **“a”** is then removed and printed on the console. **“b”** becomes the first bunker now.

Input	Output

`50`
`b 10 15 20 30`
`c 100`
`a 65`
`Bunker Revision`
	`b -> 20, 30`
	`c -> Empty`