# Softuniada 2019

## Undefined

*Have you ever heard of blockchain? Well, even if you didn’t it is not a problem. In blockchain, its all about mining blocks. Mining a block is done by 2 nodes – each node is a type of business, but the 2 nodes (businesses) must have the same owner, and they must be connected only to each other.*

You will receive N – an **integer**, which is the **amount** of **business owners**.

On the next **N lines** you will receive the owner’s **initial** – a **letter** from the **alphabet**, and his **businesses** – which, will be **integers** – each **integer**, representing the corresponding business’s **net worth**.

If **2 businesses** (a **pair** of **businesses**) are connected **ONLY** to **each other** and they have the **SAME owner**, they **WILL** **mine** a **block**. That block will have a **value** – equal to the **absolute value** of the **difference** between the 2 businesses’ net worth.

You must generate a network of business owners and **pairs** of **businesses** in which you **mine** the **blocks** with the **highest summed up value**. However, note that, **NO business** should remain **disconnected**.

### Input

The input will consist of several lines:

* On the **first** input line you will receive **N** – the **amount** of **business owners**.
* On the next **N lines** you will receive each owner’s initial and businesses in the following format:

{owner} -> {business1}, {business2}, {business3}...

### Output

As output:

* You must print the owners, with each of their business pairs, in the following format:

{owner} | {businessPair1First} <-> {businessPair1Second}, {businessPair2First}...

* + Each owner must be printed on a **new line**.
  + The **owners** should be in **order** of **addition**.
  + The **businesses** should be **ordered** by **mined block value** in **descending order**.
  + If an owner **does not have** any pairs, you should just print "none".
* You must print the **leftover connections** (the businesses, that **did not mine** any **blocks**), if there are **any**, in the following format:

{owner}{business} <-> {otherOwner}{otherBusiness}

* + The **leftover connections** must be **ordered** by the **sum** of each **2 businesses’** **net worth**, in **descending order**.
* You must print the **total mined block value**.

### Constraints

* The integer **N – count** of **owners** will be in **range [0, 25]**.
* The **businesses**’ **net worth** will be integers in **range** **[0, 100000]**.
* Each **owner** may be **given** **up** to **1000 businesses**.
* Allowed time / memory: 100ms / 16MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 3  A -> 60, 120, 40, 30  B -> 300, 4  C -> 50, 200, 220, 20 | A | 120 <-> 30, 60 <-> 40  B | 300 <-> 4  C | 220 <-> 20, 200 <-> 50  756 |  |
| 3  A -> 60, 120, 40, 30  B -> 300, 4, 4  C -> 50, 200, 220, 20, 5 | A | 120 <-> 30, 60 <-> 40  B | 300 <-> 4  C | 220 <-> 5, 200 <-> 20  B4 <-> C50  801 | Notice how we have 2 more elements, one at B and one at 5 that are left-overs, after the pairs have been generated.  We just pair them together and print the other pairs in the network, so that we mine the maximum block value. |
| 3  A -> 60, 120, 40, 30  B -> 300, 4, 4  C -> 50, 200, 220, 20 | A | 120 <-> 30  B | 300 <-> 4  C | 220 <-> 20, 200 <-> 50  B4 <-> A60  B4 <-> A40  736 | When you don’t have another leftover element with which to pair one, you will need to **ruin** a **business pair**, and you must ruin the one that will bring you the least money, so that the network remains with the highest mined bock value. |