# Little Alchemy

Professor X. has finally come up with an idea how to become rich. He designed a special **acid**, which applied in a precise amount, transforms **stone** into pure **gold**. Calculating how much acid is needed for each stone is hard, that’s why Professor X has **labeled each stone with a number**, showing how many **doses** it needs to turn into the valuable metal.

The stones are laid one after another on a table and every time he takes **the first (leftmost) stone**, he applies acid on it and returns it **on the (right) end** of the sequence. For example, he starts with stones **2 3 4**. On the first turn he will take **2**, apply acid and change the label on it to **1** (because there is only one dose left to be applied). Then, he places it on the **end of the sequence** and continue doing so.

**2 3 4 -> 3 4 1 -> 4 1 2 -> 1 2 3 …**

When the required amount of acid is applied on a stone **(label becomes 0)**, it is moved from the table to the **storage** – a secret place, where Professor X keeps the gold. Because oxygen can revert the effect of the acid, he keeps the storage **air free**.

Unfortunately, there are some **air** **leaks** into the storage, which make the gold **pieces turn back into stones**. Every time there is a leak, the **last** **piece that got into the storage gets damaged** and the Professor has to calculate again the acid doses he needs for it to turn in back into gold. After labeling the stone again, he returns it back to the table - **at the end** of the sequence.

### Input

* On the first line of the input you will receive the **sequence of the stones** the Professor has at the beginning, each represented by an **integer** **– its label**. The stones will be separated **by space**.
* Afterwards you will get a series of commands of the following types:
* **"Apply acid {n}"** – Prof. X applies **n** doses of acid on **n** number of stones and moves each at the **end**. If there are **no** stones at the table, **ignore** the command.
* **"Air Leak {m}"** – there is a leak in the storage, the **last piece gets damaged** and must be treated with **n** more doses to turn into gold again. If the storage is **empty**, **ignore** the command.
* "**Revision**" – end of input.

### Output

After receiving the "**Revision**" command you should print

* On the first line - all **stones** left, **in their current order on the table**
* On the second line - the amount of **gold pieces in the storage**

### Constrains

* Each of the integers in the input will be in the range **[1…3000]**.
* The sequence will consist of **[1…10000]** integers.
* **All data** must be processed **by order of input**.
* Allowed time/memory: **200ms/16MB**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 1 4 5  Apply acid 2  Apply acid 1  Air leak 5  Apply acid 3  Revision | 5 3 4  2 | **First** and **second** stones get **1** dose of acid, so the first **turns into gold** and the second is put at the end of the sequence with **decremented** label: **1 4 5 1.**  **First** stone turns into gold: **4 5 1.**  Air leaks into the storage and the l**ast piece of gold** transforms into **stone** with label "**5**": **4 5 1 5.**  **First**, **second** and **third** stones get 1 dose of acid and only the **second** **turns into gold**: **5 3 4.**  At the end there are **2** pieces of **gold** in the **storage**. |
| 5 5 1 2 5  Apply acid 2  Air leak 5  Apply acid 1  Revision | 2 5 4 4  1 | **1 2 5 4 4**  **1 2 5 4 4** (There is **nothing** in the storage, so the command is **ignored**)  **2 5 4 4** |