# Programming Fundamentals with Python: Exam Preparation

## Guinea Pig

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2031#0**](https://judge.softuni.org/Contests/Practice/Index/2031#0)**.**

*Merry has a guinea pig named Puppy, whom she loves very much. Every month she goes to the nearest pet store and buys him everything he needs – food, hay, and cover.*

On the **first three lines**, you will receive **the quantity of food**, **hay**, and **cover**, which Merry buys for a **month (30 days)**. On the **fourth line**, you will receive the **guinea pig's weight**.

**Every day** Puppy eats **300 gr of food**. **Every** **second** day Merry **first feeds the pet**, then gives it a **certain amount of hay** **equal to** **5%** of the rest of the **food**. On **every** **third** day, Merry puts Puppy **cover** with **a quantity of** **1/3** of its **weight**.

**Calculate** whether the quantity of **food, hay, and cover**, will be enough for a **month**.

**If Merry runs out of food, hay, or cover, stop the program!**

## Input

* **On the first line – quantity food in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**
* **On the second line – quantity hay in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**
* **On the third line – quantity cover in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**
* **On the fourth line – guinea's weight in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**

## Output

* If the food, the hay, and the cover are enough, print:
  + **"Everything is fine! Puppy is happy! Food: {excessFood}, Hay: {excessHay}, Cover: {excessCover}."**
* If one of the things is not enough, print:
  + **"Merry must go to the pet store!"**

**The output values must be formatted to the second decimal place!**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10  5  5.2  1 | Everything is fine! Puppy is happy! Food: 1.00, Hay: 1.10, Cover: 1.87. |
| You receive food – **10000**, hay – **5000**, cover – **5200**, weight – **1000** (in grams).  On the first day, Merry gives Puppy 300gr food – 9700gr food left.  On the second day, the food left is **9400gr**, so the needed hay is **9400 \* 5% = 470**,and thehay left is **4530.**  On the third day, the cover left is **4866.67,** and the food left is **9100**,and so on.  On the last day, Merry has: food – 1.00, hay – 1.10, and cover – 1.87. | |
| 1  1.5  3  1.5 | Merry must go to the pet store! |
| 9  5  5.2  1 | Merry must go to the pet store! |

## Shoot for the Win

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2305#1**](https://judge.softuni.org/Contests/Practice/Index/2305#1)**.**

Write a program that helps you keep track of your **shot targets**. You will receive a **sequence with integers**, separated by a single space, representing targets and their value. Afterward, you will be receiving indices until the **"End"** command is given, and you need to print the **targets** and the **count of shot targets**.

Every time you receive an **index**, you need to shoot the target on that index, **if it is possible**.

Every time you **shoot a target**, its value becomes **-1, and it is considered shot**. Along with that, you also need to:

* **Reduce** all the other **targets**, which have **greater values** than your **current** target, **with its value**.
* **Increase** all the other **targets**, which **have less than or equal** value to the **shot target**, **with its value.**

**Keep in mind that you can't shoot a target, which is already shot.** **You also can't increase or reduce a target, which is considered shot.**

When you receive the **"End"** command, print the targets in their current state and the **count of shot targets** in the following format:

**"Shot targets: {count} -> {target1} {target2}… {targetn}"**

### Input / Constraints

* On the **first line** of input, you will receive a **sequence** of **integers**, **separated** by **a single space – the targets sequence**.
* On the **following lines**, until the **"End"** command, you be receiving **integers** each on a single line – **the index of the target to be shot.**

### Output

* The format of the output is described above in the problem description.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 24 50 36 70  0  4  3  1  End | Shot targets 3 -> -1 -1 130 -1 | First, we shoot the target on index 0. It becomes equal to -1, and we start going through the rest of the targets. Since 50 is more than 24, we reduce it to 26 and 36 to 12 and 70 to 46. The sequence looks like that:  **-1 26 12 46**  The following index is invalid, so we don't do anything. Index 3 is valid, and after the operations, our sequence should look like that:  **-1 72 58 -1**  Then we take the first index with value 72, and our sequence looks like that:  **-1 -1 130 -1**  Then we print the result after the **"End"** command. |
| 30 30 12 60 54 66  5  2  4  0  End | Shot targets: 4 -> -1 120 -1 66 -1 -1 |  |

## Inventory

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2028#2**](https://judge.softuni.org/Contests/Practice/Index/2028#2)**.**

*As a young traveler, you gather items and craft new items.*

### Input / Constraints

You will receive a journal with some collecting items, separated with a comma and a space (**", "**). After that, until receiving "Craft!" you will be receiving different commands split by **" - "**:

* "Collect - {item}" - you should add the given item to your inventory. If the item already **exists**, you should **skip** this line.
* "Drop - {item}" - you should remove the item from your inventory **if it exists**.
* "Combine Items - {old\_item}:{new\_item}" - you should check if the **old item exists**. If so, **add** the new item **after** the old one. Otherwise, **ignore** the command.
* "Renew – {item}" – if the given item exists, you should change its position and **put it last** in your inventory.

### Output

After receiving "Craft!" print the items in your inventory, separated by **", "**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Iron, Wood, Sword  Collect - Gold  Drop - Wood  Craft! | Iron, Sword, Gold |
| **Input** | **Output** |
| Iron, Sword  Drop - Bronze  Combine Items - Sword:Bow  Renew - Iron  Craft! | Sword, Bow, Iron |