OS Planning

You are hired to create a program that schedules the work of a OS and avoids tasks that could harm it.

On the **first line** you will be given some **tasks** as **integer values**, separated by **comma and space ", "**. On the **second line** you will be given some **threads as integer values**, separated by a single space**.** On the **third line**, you will receive the **integer value** of a **task** that you need to **kill**. Your job is to **stop the work of the OS as soon as you get to this task**, otherwise your OS will crash. The **thread** that gets **first** to this task, **kills it**.

The **OS works** in the following way:

- It takes the **first given thread value** and the **last given task value.**
- If the **thread value** is **greater** than or **equal** to the **task** value, **the task and thread get removed**.
- If the **thread value** is **less** than the **task value**, the **thread** gets **removed**, but the **task remains**.

After you finish the needed task, print on a single line:

`"Thread with value {thread} killed task {taskToBeKilled}"`

Then print the remaining threads (**including** the **one that killed** the **task**) starting from the first on a single line, separated by a single space.

Input

- **On the first line** you will receive the **tasks**, separated by **", "**.
- **On the second line** you will the **threads**, separated by a single space.
- **On the third line**, you will receive a **single integer** – **value of the task** to be killed.

Output

- Print the **thread that killed the task** and **task itself** in the format given above.
- Print the **remaining threads** starting **from the first** on a single line, separated by a single space.

Constraints

- The needed task will always be with a **unique** value
- You will **always have enough threads to get to the needed task**

Examples

Input	Output	Comment

20, 23, 54, 34, 90 150 64 20 34 54	Thread with value 20 killed task 54 20 34	First, thread with value 150 is taken and the task with value 90. The thread has bigger value, so both thread and task get remove. Next, the thread 64 finishes task 34 and both get removed. Then thread 20 gets to task 54 and kills it.
33, 12, 15, 40, 45, 60 30 20 53 67 84 90 40	Thread with value 90 killed task 40 90	Thread 30 takes task 60, but it the task has greater value, so the thread gets removed. Then thread 20 takes task 60 and the same happens – thread get removed. Then the same happens with thread 53. After that, thread 67 takes task 60 and finishes it. Then thread 84 finishes task 45.Finally, thread 90 gets to task 40, which should be killed and the program stops.