01.Rubber Duck Debuggers

Rubber Duck Debugging is a type of debugging where you place a rubber duck on your desk and explain to it your code line by line. You gathered a few programmers and gave them a task and judging by the time it took them to write the code, you reward them with a type of rubber ducky.

Learn more about Rubber Duck Debugging after your exam from here.

You will be given **two sequences of integers**. The first one represents **the time it takes a programmer to complete a single task**. The second one represents the **number of tasks** you've given to your programmers.

Your task is to **count** how many **rubber ducks of each type** you've given to your programmers.

While you have values in the sequences, you need to start from the **first value of the programmers time's sequence** and **multiply** them by **the last value of the tasks' sequence**:

- If the calculated time is **between** any of the **time ranges below, you give the corresponding ducky** and **remove the programmer time's value and the tasks' value**.
- If the calculated time goes **above** the **highest range**, **decrease the number of the task's value by 2**. Then, **move the programmer time's value** to the **end** of its sequence, and continue with the next operation.

Rubber Ducky type	Time needed to earn it
Darth Vader Ducky	0 - 60
Thor Ducky	61 – 120
Big Blue Rubber Ducky	121 - 180
Small Yellow Rubber Ducky	181 - 240

Your task is considered done when the **sequences are empty**.

Input

- The first line will represent each **programmer's time** to complete a single task – **integers**, separated by a **single space.**
- The second line will represent the **number of tasks** that should be completed – **integers**, separated by a **single space**.

Output

- On the **first** line, you output:
  - `"Congratulations, all tasks have been completed! Rubber ducks rewarded:"`
- On the next **4 lines,** you output the **type** and **number** of ducks given, **ordered** like in the **table**:

  -
    ```
    "Darth Vader Ducky: {count}
    Thor Ducky: {count}
    Big Blue Rubber Ducky: {count}
    Small Yellow Rubber Ducky: {count}"
    ```

Constraints

- All the given numbers will be valid **integers** in the range **[1, 1000].**
- There will be **no negative inputs**.
- **The number of values in both sequences will always be equal**.

Examples

Input	Output

`10 15 12 18 22 6`
`12 16 5 6 9 1`
	`Congratulations, all tasks have been completed! Rubber ducks rewarded:`
	`Darth Vader Ducky: 1`
	`Thor Ducky: 3`
	`Big Blue Rubber Ducky: 1`
	`Small Yellow Rubber Ducky: 1`

Comment

1) 10 15 12 18 22 6
   12 16 5 6 9 1
   10 * 1 = 10
   Give a Darth Vader Ducky. Remove the programmer time's value (10) and the task's value (1)
2) 15 12 18 22 6
   12 16 5 6 9
   15 * 9 = 135
   Give a Big Blue Rubber Ducky. Remove the programmer time's value (15) and the task's value (9)
3) 12 18 22 6
   12 16 5 6
   12 * 6 = 72
   Give a Thor Ducky. Remove the programmer time's value (12) and the task's value (6)
4) 18 22 6
   12 16 5
   18 * 5 = 90
   Give a Thor Ducky. Remove the programmer time's value (18) and the task value (5)
5) 22 6
   12 16
   22 * 16 = 352
   Return the programmer time's value (22) at the end of the sequence and decrease the task's value (16) by 2 -> it becomes 14
6) 6 22
   12 14
   6 * 14 = 84
   Give a Thor Ducky. Remove the programmer time's value (6) and the task value (14)

7) 22
   12
   22 * 12 = 264
   Return the programmer time's value (22) at the end of the sequence and decrease the task's value (12) by 2 -> it becomes 10
8) 22
   10
   22 * 10 = 220
   Give a Small Yellow Rubber Ducky. Remove the programmer time's value (22) and the task's value (10). There are no more elements in the sequences, so we print the needed output and end the program.

Input	Output

`2 55 17 31 23`
`7 5 17 4 27`
	`Congratulations, all tasks have been completed! Rubber ducks rewarded:`
	`Darth Vader Ducky: 1`
	`Thor Ducky: 0`
	`Big Blue Rubber Ducky: 2`
	`Small Yellow Rubber Ducky: 2`

Comment

1) 2 * 27 = 54 -> Give a Darth Vader Ducky.
2) 55 * 4 = 220 -> Give a Small Yellow Rubber Ducky.
3) 17 * 17 = 289 -> Programmer time's value is sent to the end and the task's value is decreased by 2
4) 31 * 15 = 465 -> Programmer time's value is sent to the end and the task's value is decreased by 2
5) 23 * 13 = 299 -> Programmer time's value is sent to the end and the task's value is decreased by 2
6) 17 * 11 = 187 -> Give a Small Yellow Rubber Ducky.
7) 31 * 5 = 155 -> Give a Big Blue Rubber Ducky.
8) 23 * 7 = 161 -> Give a Small Yellow Rubber Ducky.