## Problem 1 – Black Flag

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/3950/programming-fundamentals-with-csharp-january-2023)

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/1773#0)

*Pirates are invading the sea, and you're tasked to help them plunder*

Create a program that checks if **target plunder** is **reached**. First, you will receive how many **days** the pirating lasts. Then you will receive how much the pirates **plunder for a day**. Last you will receive the **expected plunder** at the end.

Calculate how much **plunder** the pirates manage to **gather**. Each **day** they gather the **plunder**. Keep in mind that they attack more ships every third day and add additional plunder to their total gain, which is **50% of the daily plunder**. Every **fifth day** the pirates encounter a warship, and after the battle, they **lose 30%** of their **total plunder**.

If the gained plunder is **more or equal** to the target, print the following:

**"Ahoy! {totalPlunder} plunder gained."**

If the gained plunder is **less** than the target. Calculate the **percentage left** and print the following:

**"Collected only {percentage}% of the plunder."**

Both numbers should be **formatted** to the **2nd decimal place**.

### Input

* On the **1st line,** you will receive the **days** of the plunder – an **integer number** in the range [0…100000].
* On the **2nd line,** you will receive the **daily plunder** – an **integer number** in the range [0…50].
* On the **3rd line,** you will receive the **expected plunder** – a **real number** in the range [0.0…10000.0].

### Output

* In the end, print whether the plunder **was successful** or **not,** following the format **described above**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 40 100 | Ahoy! 154.00 plunder gained. |
| **Comments** | |
| The days are 5, and the daily plunder is 40. On the third day, the total plunder is 120, and since it is a third day, they gain an additional 50% from the daily plunder, which adds up to 140. On the fifth day, the plunder is 220, but they battle with a warship and lose 30% of the collected cargo, and the total becomes 154. That is more than expected. | |
|  | |
| 10  20  380 | Collected only 36.29% of the plunder. |

## Problem 2 – Shoot for the Win

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/3950/programming-fundamentals-with-csharp-january-2023)

Submit your solutions in the SoftUni Judge system [here](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 |  |

## Problem 3 – Numbers

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/3950/programming-fundamentals-with-csharp-january-2023)

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/2474#2)

Write a program to **read a sequence of integers** and find and print the **top 5** numbers **greater than the average** value in the sequence, sorted in descending order.

### Input

* Read from the console a single line holding **space-separated integers**.

### Output

* Print the above-described numbers on a single line, space-separated.
* If **less than 5 numbers** hold the property mentioned above, **print less** than 5 numbers.
* Print **"No"** if no numbers hold the above property.

### Constraints

* All input **numbers** are integers in the **range** [-1 000 000 … 1 000 000].
* The **count of numbers** is in the **range** [1…10 000].

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10 20 30 40 50 | 50 40 | Average number = 30.  Numbers greater than 30 are: {40, 50}.  The top 5 numbers among them in descending order are: {50, 40}.  Note that we have only 2 numbers, so all of them are included in the top 5. |
| 5 2 3 4 -10 30 40 50 20 50 60 60 51 | 60 60 51 50 50 | Average number = 28.08.  Numbers greater than 28.08 are: {30, 40, 50, 50, 60, 60, 51}.  The top 5 numbers among them in descending order are: {60, 60, 51, 50, 50}. |
| 1 | No | Average number = 1.  There are no numbers greater than 1. |
| -1 -2 -3 -4 -5 -6 | -1 -2 -3 | Average number = -3.5.  Numbers greater than -3.5 are: {-1, -2, -3}.  The top 5 numbers among them in descending order are: {-1, -2, -3}. |