Programming Fundamentals with Python: Exam Preparation

1. Counter-Strike

Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2305#0.

Write a program that keeps track of every won battle against an **enemy**. You will receive **initial energy**. Afterward, you will start receiving the **distance** you need **to reach an enemy** until the **"End of battle"** command is given or you **run out of energy**.

The **energy** you need to reach an enemy is **equal to the distance you receive**. Each time you reach an enemy, you **win** a battle, and your **energy is reduced**. Otherwise, if you don't have **enough energy** to reach an enemy, **end the program** and **print**: **"Not enough energy! Game ends with {count} won battles and {energy} energy"**.

Every **third won battle** increases **your energy with the value of your current count of won battles**.

Upon receiving the **"End of battle"** command**,** print the **count of won battles** in the following format:

"Won battles: {count}. Energy left: {energy}"

Input / Constraints

- On the **first line,** you will receive **initial energy** – an **integer [1-10000]**.
- On the **following lines,** you will be receiving the **distance** of an enemy – an **integer [1-10000]**

Output

- The description contains the proper output messages for each case and the format they should be printed.

Examples

Input	Output	Comments
100 10 10 10 1 2 3 73 10	Not enough energy! Game ends with 7 won battles and 0 energy	The initial energy is 100. The first distance is 10, so we subtract 10 from 100, and we consider this a **won** battle. We are left with 90 energy. Next distance – 10, and 80 energy left. Next distance – 10, 3 won battles and 70 energy, but since we have 3 won battles, we increase the energy with the current count of won battles, in this case – **3, and it becomes 73**. The last distance we receive – **10** is unreachable since we have **0** energy, so we print the appropriate message, and the program ends.
200 54 14 28 13 End of battle	Won battles: 4. Energy left: 94	

2. The Lift

Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2517#1.

Write a program that **finds a place for the tourist on a lift.**

Every wagon should have **a maximum of 4 people on it**. If a wagon is full, you should direct the people to **the next one with space** available.

Input

- **On the first line,** you will receive **how many people** are waiting to get **on the lift**
- **On the second line**, you will receive the **current state of the lift separated by a single space: " "**.

Output

When there is no more available space left on the lift, or there are **no more people in the queue**, you should print on the console the final state of the lift's wagons separated by **" "** and one of the following messages:

- If there are no more people and the lift have empty spots, you should print:

  "The lift has empty spots!

  {wagons separated by ' '}"

- If there are still people in the queue and no more available space, you should print:

  "There isn't enough space! {people} people in a queue!

  {wagons separated by ' '}"

- If the lift is full and there are no more people in the queue, you should print only the wagons separated by **" "**

Examples

Input	Output
15 0 0 0 0	The lift has empty spots! 4 4 4 3
Comment	
First state - 4 0 0 0 -> 11 people left Second state – 4 4 0 0 -> 7 people left Third state – 4 4 4 0 -> 3 people left	
Input	**Output**
20 0 2 0	There isn't enough space! 10 people in a queue! 4 4 4
Comment	
First state - 4 2 0 -> 16 people left Second state – 4 4 0 -> 14 people left Third state – 4 4 4 -> 10 people left, but there're no more wagons.	

3. Numbers

Submit your solutions in the SoftUni judge system at 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}.