# SoftUni Reception

Every day thousands of students pass by the reception at SoftUni with different questions to ask and the employees have to help everyone by providing all the information and to answer all of the questions.

There are **3 employees** working on the reception all day. Each of them can handle **different number** **of students** **per hour**. Your task is to **calculate how much time** it will take to **answer all the questions** of given number of students.

First you will receive 3 lines with integers, representing **count of students that each of the employee can help per hour.** On the next line you will receive **students count as a single integer**.

**Every forth hour all of the employees have a break, so they don’t work for a hour.** This is the only break for the employees, because they don`t need rest, nor have a personal life. Calculate the time needed to answer all the student's questions and print it in the following format: "Time needed: {time}h."

## Input / Constraints

* On first three lines - **each employee efficiency** - integer in range **[1 - 100]**
* On the fourth line - **students count** – integer in range **[0 – 10000]**
* Input will always be valid and in the range specified

## Output

* Print a single line: "Time needed: {time}h."
* Allowed working **time** / **memory**: **100ms** / **16MB**

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 5  6  4  20 | Time needed: 2h. | All employees can answer 15 students per hour. After the first hour there are 5 students left to be answered.  All students will be answered in the second hour. |
| 1  2  3  45 | Time needed: 10h. | All employess can answer 6 students per hour. In the first 3 hours they have answered 6 \* 3 = 18 students. Then they have a break for an hour.  After the next 3 hours there are  18 + 6 \* 3 = 36 answered students.  After the break for an hour, there are only 9 students to answer.  So in the 10th hour all of the student's questions would be answered. |
| 3  2  5  40 | Time needed: 5h. |  |

|  |  |  |
| --- | --- | --- |
| **Input JavaScript** | **Output** | **Comment** |
| ['5','6','4','20'] | Time needed: 2h. | All employees can answer 15 students per hour. After the first hour there are 5 students left to be answered.  All students will be answered in the second hour. |
| ['1','2','3','45'] | Time needed: 10h. | All employess can answer 6 students per hour. In the first 3 hours they have answered 6 \* 3 = 18 students. Then they have a break for an hour.  After the next 3 hours there are  18 + 6 \* 3 = 36 answered students.  After the break for an hour, there are only 9 students to answer.  So in the 10th hour all of the student's questions would be answered. |
| ['3','2','5','40'] | Time needed: 5h. |  |

# 02. Array Modifier

You are given **an array with integers**. Write a program to **modify the elements** after **receive the commands** “**swap**”, “**multiply**” or “**decrease**”.

* “swap {index1} {index2}” take **two elements** and **swap their places**.
* “multiply {index1} {index2}” take **element at the 1st index** and **multiply** **it** **with element at 2nd index**. **Save the product at the 1st index.**
* “decrease” **decreases** **all elements** in the array **with 1**.

## Input

On the **first input line** you will be given **the initial array values** separated by a single space.

On the **next lines** you will receive commands **until** you receive the **command “end”**. The **commands are** as follow:

* “swap {index1} {index2}”
* “multiply {index1} {index2}”
* “decrease”

## Output

**The output** should be printed on the console and consist **element** **of the** **modified array** – **separated by “, “**(**comma and single space**).

## Constraints

* Commands will be: “**swap**”, “**multiply**”, “**decrease**” and “**end**”
* **Elements of the array** will be **integer numbers** in the range **[-231...231]**
* **Count of the array elements** will be in the range **[2...100]**
* **Indexes** **will be always in the range of the array**

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 23 -2 321 87 42 90 -123  swap 1 3  swap 3 6  swap 1 0  multiply 1 2  multiply 2 1  decrease  end | 86, 7382, 2369942, -124, 41, 89, -3 | 23 -2 321 87 42 90 -123 – initial values  swap 1(-2) and 3(87) ▼  23 87 321 -2 42 90 -123  swap 3(-2) and 6(-123) ▼  23 87 321 -123 42 90 -2  swap 1(87) and 0(23) ▼  87 23 321 -123 42 90 -2  multiply 1(23) 2(321) = 7383 ▼  87 7383 321 -123 42 290 -2  multiply 2(321) 1(7383) = 2369943 ▼  87 7383 2369943 -123 42 90 -2  decrease – all - 1 ▼  86 7383 2369942 -124 41 89 -3 |
| 1 2 3 4  swap 0 1  swap 1 2  swap 2 3  multiply 1 2  decrease  end | 1, 11, 3, 0 |  |

|  |  |  |
| --- | --- | --- |
| **Input JavaScript** | **Output** | **Comments** |
| [  '23 -2 321 87 42 90 -123',  'swap 1 3',  'swap 3 6',  'swap 1 0',  'multiply 1 2',  'multiply 2 1',  'decrease',  'end'  ] | 86, 7382, 2369942, -124, 41, 89, -3 | 23 -2 321 87 42 90 -123 – initial values  swap 1(-2) and 3(87) ▼  23 87 321 -2 42 90 -123  swap 3(-2) and 6(-123) ▼  23 87 321 -123 42 90 -2  swap 1(87) and 0(23) ▼  87 23 321 -123 42 90 -2  multiply 1(23) 2(321) = 7383 ▼  87 7383 321 -123 42 290 -2  multiply 2(321) 1(7383) = 2369943 ▼  87 7383 2369943 -123 42 90 -2  decrease – all - 1 ▼  86 7383 2369942 -124 41 89 -3 |
| [  '1 2 3 4',  'swap 0 1',  'swap 1 2',  'swap 2 3',  'multiply 1 2',  'decrease',  'end'  ] | 1, 11, 3, 0 |  |

# 03. Numbers

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

## Input

Read from the console a single line holding space separated number.

## Output

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

## Constraints

All input numbers are integers in range [-1 000 000 … 1 000 000]. The count of numbers is in 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 20.078 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}. |

|  |  |  |
| --- | --- | --- |
| **Input JavaScript** | **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 20.078 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}. |