# JS Fundamentals Mid Exam Preparation

### Problem 1. Bonus Scoring System

**Link:** <https://judge.softuni.org/Contests/Practice/Index/2028#0>

Create a program that calculates **bonus points** for each **student**, for a certain **course**. On the first line, you are going to receive **the count of the students** for this course. **On the second line**, you will receive **the count of the lectures** in the course. Every course has **an additional bonus**. You are going to receive it **on the third line**. On the next lines, you will be receiving the **count of attendances** **for each student**.

The bonus is calculated with the following **formula**:

**{total bonus} = {student attendances} / {course lectures} \* (5 + {additional bonus})**

Find the student with the **maximum bonus** and print him/her, along with **his attendances** in the following format:

**"Max Bonus: {maxBonusPoints}."**

**"The student has attended {studentAttendances} lectures."**

Round the bonus points at the end to **the nearest bigger number**.

### Input / Constrains

* On the **first line** you are going to receive the count of the students – an integer number in the range [0…50]
* On the **second line** you are going to receive the count of the lectures – an integer number in the range [0...50].
* On the **third line** you are going to receive **the initial bonus** – an integer number in the range [0….100].
* **On the next lines**, you will be receiving the **attendances of each student**.
* There will **never** be **students with equal bonuses**.

### Output

* Print the maximum bonus points along with the attendances of the given student, **rounded** to the nearest **bigger** number, scored by a student in this course in the format described above.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  25  30  12  19  24  16  20 | Max Bonus: 34.  The student has attended 24 lectures. |
| **Comments** | |
| First, we receive the **number of students** enrolled in the course – **5**. The total count of the lectures is **25** and the initial bonus is **30**. Then we calculate the bonus of the student with 12 attendances, which is **16.8**. We continue calculating **each of the student’s bonuses**. The one **with 24 attendances** has the **highest bonus – 33.6 (34 rounded)**, so we print the appropriate message on the console. | |
| 10  30  14  8  23  27  28  15  17  25  26  5  18 | Max Bonus: 18.  The student has attended 28 lectures. |

### JS Input / Output

|  |  |
| --- | --- |
| **Input** | **Output** |
| [  '5', '25', '30',  '12', '19', '24',  '16', '20'  ] | Max Bonus: 34.  The student has attended 24 lectures. |
| **Comments** | |
| First, we receive the **number of students** enrolled in the course – **5**. The total count of the lectures is **25** and the initial bonus is **30**. Then we calculate the bonus of the student with 12 attendances, which is **16.8**. We continue calculating **each of the student’s bonuses**. The one **with 24 attendances** has the **highest bonus – 33.6 (34 rounded)**, so we print the appropriate message on the console. | |
| [  '10', '30', '14', '8',  '23', '27', '28', '15',  '17', '25', '26', '5',  '18'  ] | Max Bonus: 18.  The student has attended 28 lectures. |

### Problem 2. Shoot for the Win

Link: <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 single space, representing targets and their value. Afterwards, 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 possiblie**.

Everytime 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**.
* All the **targets**, which **have less than or equal** value to the **shot target**, you need to **increase** **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 **next 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 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 next 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 |  |

### JS Input / Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| (["24 50 36 70",  "0",  "4",  "3",  "1",  "End"]) | Shot targets 3 -> -1 -1 130 -1 | First we shoot 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 next 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

Link: <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 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 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}. |

## JS Input / Output

|  |  |  |
| --- | --- | --- |
| **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}. |