**JS Syntax Fundamentals – More Exercises**

Problems with exercise and homework for the ["JS Front-End" Course @ SoftUni.](https://softuni.bg/modules/132)

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/3787/JS-Syntax-Fundamentals-More-Exercises>

1. **\*Validity Checker**

Write a program that receives a total of 4 parameters in the format **x1, y1, x2, y2.** Check if the distance between each point (**x, y**) and the beginning of the Cartesian coordinate system (**0, 0**) is **valid**. A distance between two points is considered **valid** if it is an **integer value**.

**Note:** You can use the following formula to help you calculate the distance between the points (**x1, y1**) and (**x2, y2**).

The order of comparisons should always be first **{x1, y1}** to **{0, 0}**, then **{x2, y2}** to **{0, 0}** and finally **{x1, y1}** to **{x2, y2}**.

In case a distance is valid, print: **`**{x1, y1} to {x2, y2} is valid**`**

If the distance is invalid, print: **`**{x1, y1} to {x2, y2} is invalid**`**

The **input** consists of two points given as **4 numbers**.

For each comparison print either **`**{x1, y1} to {x2, y2} is valid**`** if the distance is valid, or **`**{x1, y1} to {x2, y2} is invalid**`** if it is invalid.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 0, 0, 4 | {3, 0} to {0, 0} is valid  {0, 4} to {0, 0} is valid  {3, 0} to {0, 4} is valid |
| 2, 1, 1, 1 | {2, 1} to {0, 0} is invalid  {1, 1} to {0, 0} is invalid  {2, 1} to {1, 1} is valid |

## \*Words Uppercase

Write a program that **extracts all words** from a passed-in string and converts them to **upper case**. The extracted words in the upper case must be printed on a single line separated by **", "**.

The **input** comes as a single string argument - the text to extract and convert words from.

The **output** should be a single line containing the converted string.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 'Hi, how are you?' | HI, HOW, ARE, YOU | 'hello' | HELLO |

### Hints

* You may need to use a [Regular Expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions) or check for all delimiters that can be found in a sentence (ex. ",", " ", "!", "?" and so on).

## \*Calculator

Write a **function** that receives 3 parameters: a **number**, an **operator** (string), and **another number**.

The **operator** can be: **'+', '-', '/', '\*'.** Print the result of the calculation on the console formatted to the **second decimal** point.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5,  '+',  10 | 15.00 |
| 25.5,  '-',  3 | 22.50 |

## \*Gladiator Expenses

As a gladiator, Peter has to repair his broken equipment when he loses a fight. His equipment consists of a helmet, sword, shield, and armor. You will receive Peter`s **lost fights count**.

* Every **second** lost game, his helmet is broken.
* Every **third** lost game, his sword is broken.
* When both **his sword and helmet are broken** in the same lost fight, his **shield also breaks**.
* **Every** **second time**, when his shield brakes, his **armor** also needs to be repaired.

You will receive the price of each item in his equipment. Calculate his expenses for the year for renewing his equipment.

### Input / Constraints

You will receive 5 parameters to your function:

* The first parameter - **lost fights count** - is an integer in the range **[0, 1000]**.
* The second parameter - **helmet price** - is the floating-point number in the range **[0, 1000]**.
* The third parameter - **sword price** - is the floating-point number in the range **[0, 1000]**.
* The fourth parameter - **shield price** - is the floating-point number in the range **[0, 1000]**.
* The fifth parameter - **armor price** - is the floating-point number in the range **[0, 1000]**.

### Output

* As output you must print Peter`s total expenses for new equipment rounded to the second decimal point: **"Gladiator expenses: {expenses} aureus"**
* Allowed working **time** / **memory**: **100ms** / **16MB**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 7,  2,  3,  4,  5 | Gladiator expenses: 16.00 aureus | Trashed helmet -> 3 times  Trashed sword -> 2 times  Trashed shield -> 1 time  Total: 6 + 6 + 4 = 16.00 aureus; |
| 23,  12.50,  21.50,  40,  200 | Gladiator expenses: 608.00 aureus |  |

## \*Spice Must Flow

*Spice is Love, Spice is Life. And most importantly, Spice must flow. It must be extracted from the scorching sands of Arrakis, under the constant threat of giant sandworms. To make the work as efficient as possible, the Duke has tasked you with the creation of management software.*

Write a program that calculates the **total amount** of spice that can be extracted from a source. The source has a **starting yield**, which indicates how much spice can be mined on the **first day**. After it has been mined for a day, the **yield drops** by 10, meaning on the second day it’ll produce 10 less spice than on the first, on the third day 10 less than on the second, and so on (see examples). A source is considered profitable only while its yield is **at least** 100 – when less than 100 spices are expected in a day, abandon the source.

The mining crew **consumes** 26 spices **every day** at the end of their shift and **an additional** 26 after the mine has been exhausted. Note that the workers **cannot** consume more spice than there is in storage.

When the operation is complete, print on the console on two separate lines how many **days** the mine has operated and the **total** amount of spice extracted.

### Input

You will receive a number, representing the **starting yield** of the source.

### Output

Print on the console on two separate lines how many **days** the mine has operated and the **total amount** of spice extracted.

### Constraints

* The starting yield will be a **number** within range [0…228].

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanation** |
| 111 | 2  134 | **Day 1** we extract 111 spices and at the end of the shift, the workers consume 26, leaving 85. The yield drops by 10 to 101.  **On day 2** we extract 101 spices, the workers consume 26, leaving 75. The total is 160 and the yield has dropped to 91.  **Since** the expected yield is less than 100, we abandon the source. The workers take another 26, leaving 134. The mine has operated for 2 days. |
| 450 | 36  8938 |  |