# More Exercises: Functions

Problems for more exercise for the ["PHP Fundamentals" course @ SoftUni](https://softuni.bg/trainings/2344/php-fundamentals-may-2019).

You can check your solutions in [Judge](https://judge.softuni.bg/Contests/1293/)

## Data Types

Write a program that, depending on the first line of the input, reads an **int**, **double** or **string**.

* If the data type is int, multiply the number by 2.
* If the data type is real, multiply the number by 1.5 and format it to the second decimal point.
* If the data type is string, surround the input with "$".

Print the result on the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| int  5 | 10 |
| real  2 | 3.00 |
| string  hello | $hello$ |

## Center Point

You are given the coordinates of two points on a [Cartesian coordinate system](https://en.wikipedia.org/wiki/Cartesian_coordinate_system) - **X1**, **Y1**, **X2** and **Y2**. **Create a function** that prints the point that is closest to the center of the coordinate system (0, 0) in the format (X, Y). If the points are on a same distance from the center, print only the first one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  -1  2 | (-1, 2) |

## Longer Line

You are given the coordinates of four points in the 2D plane. The first and the second pair of points form two different lines. Print the longer line in format "(X1, Y1)(X2, Y2)" starting with the point that is closer to the center of the coordinate system (0, 0) (You can reuse the function that you wrote for the previous problem). If the lines are of equal length, print only the first one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  -1  2  -5  -5  4  -3 | (4, -3)(-5, -5) |

## Tribonacci Sequence

In the **"Tribonacci" sequence**, every number is formed by the **sum of the previous 3**.

You are given a number num. Write a program that printsnumnumbers from the Tribonacci sequence, each on a new line, starting from 1. The value num will always be positive integer.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 | 1 1 2 4 |
| 8 | 1 1 2 4 7 13 24 44 |
| 2 | 1 1 |

## Multiplication Sign

You are given a number num1, num2 and num3. Write a program that finds if num1 \* num2 \* num3 (the product) is **negative**, **positive or zero**. Try to do this **WITHOUT** multiplying the 3 numbers.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  3  -1 | negative |
| 2  3  1 | positive |
| -2  0  4 | zero |

## Inventory Matcher

You will be given **three** arrays on **different** **lines**. The **first** one will contain **strings**, which will represent the **name** of **products**. **Second** one will contain **long integer numbers (**float**)** and will represent the **quantities** of the products. The **third** one will contain **float** and represents the **price** of the **product**.

After which, you will be given **names of products** on **new lines**, **until** you receive the command “done”. For each given product name print:

{name of the product} costs: {price}; Available quantity: {quantity}

**The names, prices** and **quantities** of the products are in the **same indices** in the 3 arrays.

### Input

* On the **first** **line**, you will receive an array of **strings**, which represent the **names** of the products.
* On the **second** **line,** you will receive an array of **long integer numbers (**float**)**, which represent the **quantities** of the products.
* On the **third** **line,** you will receive an array of **floats**, which represent the **prices** of the products.

### Constraints

* The **three** arrays will **always** have the **same** length.
* You will **always** receive **existing** products.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Bread Juice Fruits Lemons**  **10 50 20 30**  **2.34 1.23 3.42 1.50**  Bread  Juice  done | Bread costs: 2.34; Available quantity: 10  Juice costs: 1.23; Available quantity: 50 |
| **Oranges Apples Nuts**  **1500 5000000 2000000000**  **2.3412 1.23 3.4250**  Nuts  done | Nuts costs: 3.4250; Available quantity: 2000000000 |

### Hints

The simplest way to find the index of the element (without external libraries) will be to loop through the array

## Upgraded Matcher

For this task, you can use your solution from Inventory Matcher. You will again receive **3** **arrays** – one with **strings**, one with **long integer numbers (**float**)** and one with **floats**. Again, the **price** and **quantity** correspond to a **name**, which is located on **same** **index** as the name.

This time **only** the **arrays** containing the **names** and the array containing the **prices** will have the **same** **length**. If in the **quantities** array there is **no** **index**, which **corresponds** to the **name**, you should assume the quantity is **0**.

On top of that the products, which you receive after the arrays will contain **not** **only** a string for the **name**, but also a **long integer number (**float**)**, which is the **quantity** that must be **ordered**.

If you have **enough** **quantity**, calculate the total price by **multiplying** the ordered quantity **times** the **price** and **print it** in the following format:

{product name} x {quantity ordered} costs {total price of the order}

Format the price to the **2nd** **decimal place**. Do not forget to **decrease** the **quantity** of the product.

If you do **not** have **enough** **quantities** print:

We do not have enough {product name}

### Input

* On the **first** **line**, you will receive array of **strings**, which represent the **names** of the products.
* On the **second** **line,** you will receive array of **long integer numbers (**float**)**, which represent the **quantities** of the products.
* On the **third** **line,** you will receive array of **floats**, which represent the **prices** of the products.

### Constraints

* The **name** and **price arrays** will **always** have the **same** length.
* You will **always** receive **existing** products

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Bread Juice Fruits Lemons Beer**  **10 50 20 30**  **2.34 1.23 3.42 1.50 3.00**  Bread 10  Juice 5  Beer 20  done | Bread x 10 costs 23.40  Juice x 5 costs 6.15  We do not have enough Beer |
| **Tomatoes Onions Lemons**  **10000 2000**  **5.40 3.20 2.20**  Tomatoes 5000  Tomatoes 5000  Tomatoes 1  done | Tomatoes x 5000 costs 27000.00  Tomatoes x 5000 costs 27000.00  We do not have enough Tomatoes |