More Exercises: Lists Basics

Additional exercises for the Python Fundamentals Course @SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1726

1. Zeros to Back

Write a program that receives a **single string** (**integers separated** by a comma and space **", "**), finds all the **zeros** and moves them **to the back** without messing up the other elements. Print the resulting **integer list**

Example

Input	Output
1, 0, 1, 2, 0, 1, 3	[1, 1, 2, 1, 3, 0, 0]

2. Tic-Tac-Toe

You will receive a field of a tic-tac-toe game in **three lines** containing **numbers separated by a single space**.

Legend:

- 0 - **empty** space
- 1 - **first** player move
- 2 - **second** player move

Find out who the **winner** is. If the **first** player **wins** print **"First player won"**, otherwise if the **second** player **wins** print **"Second player won"**, otherwise print **"Draw!"**

Example

Input	Output
2 0 1 0 1 0 1 0 2	First player won
0 1 0 2 2 2 1 0 0	Second player won
1 0 2 0 1 2 1 2 0	Draw!

3. Josephus Permutation

This problem takes its name by arguably the most important event in the life of the ancient historian Josephus: according to his tale, he and his 40 soldiers were trapped in a cave by the Romans during a siege. Refusing to surrender to the enemy, they instead opted for mass suicide, with a twist: they formed a circle and proceeded to kill one man every three, until one last man was left (and that it was supposed to kill himself to end the act). Well, Josephus and another man were the last two and, as we now know every detail of the story, you may have correctly guessed that they didn't exactly follow through the original idea.

You are now to create a program that prints a **Josephus permutation**, receiving **two lines** of code (the list itself **(string** with elements separated **by a single space)** and a number **k**) as if they were in a circle and **counted out every k** places until none remained.

Example

Input	Output	Comment
1 2 3 4 5 6 7 3	[3,6,2,7,5,1,4]	[1,2,3,4,5,6,7] - initial sequence [1,2,4,5,6,7] => 3 is counted out and goes into the result [3] [1,2,4,5,7] => 6 is counted out and goes into the result [3,6] [1,4,5,7] => 2 is counted out and goes into the result [3,6,2] [1,4,5] => 7 is counted out and goes into the result [3,6,2,7] [1,4] => 5 is counted out and goes into the result [3,6,2,7,5] [4] => 1 is counted out and goes into the result [3,6,2,7,5,1] [] => 4 is counted out and goes into the result [3,6,2,7,5,1,4]

4. Battle Ships

You will be given a number **n** representing the number of **rows of the field**. On the next **n** lines you will receive **each row** of the field as a **strin**g with **numbers separated by a space**. Each number greater than zero represents a **ship** with a **health** equal to the **number value**. After that you will receive the **squares** that are being **attacked** in the format: **"{row}-{col} {row}-{col}"**. Each time a square is being attacked, if there is a ship there (number greater than 0) you should **reduce its value**. After the attacks have ended, print **how many ships** were **destroyed** (if its **value** has **reached zero**)

Example

Input	Output	Comment
3 1 0 0 1 2 0 0 0 0 3 0 1 0-0 1-0 2-1 2-1 2-1 1-1 2-1	2	States after each attack: First attack -> 1 ship destroyed 0 0 0 1 2 0 0 0 0 3 0 1 Second attack -> no ship on the square Third attack -> reduce ship health 0 0 0 1 2 0 0 0 0 2 0 1 Fourth attack -> new heath = 1 0 0 0 1 2 0 0 0 0 1 0 1 Fifth attack -> another ship destroyed 0 0 0 1

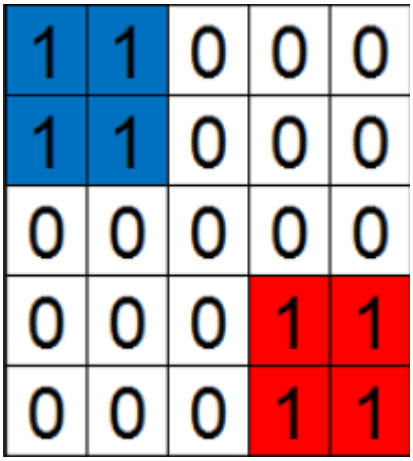
		2 0 0 0 0 0 0 1 Sixth and Seventh attack -> no ship destroyed

5. Hungry Hippos

Hungry Hippos is a tabletop game made for 2–4 players, produced by Hasbro, under the brand of its subsidiary, Milton Bradley. The idea for the game was published in 1967 by toy inventor Fred Kroll and it was introduced in 1978. The objective of the game is for each player to collect as many marbles as possible with their 'hippo' (a toy hippo model).

You will receive on the **first line** the **rows** of the matrix (**n**) and on the next **n lines** you will get **each row** of the matrix **as a string** (**zeros** and **ones** separated by a **single space**). You have to calculate how many **blocks** of food you have (**connected ones horizontally or vertically**)

Example: Here you have **2 blocks** of food



Example

Input	Output
5 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1	2
6 1 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0	1
4 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0	5