# Lab: Tuples and Sets

Problems for in-class lab for the [Python Advanced Course @SoftUni](https://softuni.bg/trainings/3848/python-advanced-september-2022).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/1832>.

## Count Same Values

You will be given **numbers separated by a space**. Write a program that **prints the number of occurrences** of each number in the format **"{number} - {count} times"**. The **number** must be **formatted** to the **first decimal point**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| -2.5 4 3 -2.5 -5.5 4 3 3 -2.5 3 | -2.5 - 3 times  4.0 - 2 times  3.0 - 4 times  -5.5 - 1 times |
| 2 4 4 5 5 2 3 3 4 4 3 3 4 3 5 3 2 5 4 3 | 2.0 - 3 times  4.0 - 6 times  5.0 - 4 times  3.0 - 7 times |

## Students' Grades

Write a program that reads students' names and their grades and adds them to the student record.

On the **first line,** you will receive **the number of students** – **N**. On the following **N** lines, you will be receiving a student's **name** and their **grade**.  
For **each student** print **all his/her grades** and finally his/her **average grade**, **formatted to the second decimal** **point** in the format: **"{student's name} -> {grade1} {grade2} ... {gradeN} (avg: {average\_grade})"**.

The **order** in which we **print** the result does not matter.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 7  Peter 5.20  Mark 5.50  Peter 3.20  Mark 2.50  Alex 2.00  Mark 3.46  Alex 3.00 | Mark -> 5.50 2.50 3.46 (avg: 3.82)  Peter -> 5.20 3.20 (avg: 4.20)  Alex -> 2.00 3.00 (avg: 2.50) |
| 4  Scott 4.50  Ted 3.00  Scott 5.00  Ted 3.66 | Ted -> 3.00 3.66 (avg: 3.33)  Scott -> 4.50 5.00 (avg: 4.75) |
| 5  Lee 6.00  Lee 5.50  Lee 6.00  Peter 4.40  Kenny 3.30 | Peter -> 4.40 (avg: 4.40)  Lee -> 6.00 5.50 6.00 (avg: 5.83)  Kenny -> 3.30 (avg: 3.30) |

## Record Unique Names

Write a program, which will take a list of **names** and print **only** the **unique** names in the list.

The **order** in which we **print** the result does not matter.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 8  Lee  Joey  Lee  Joe  Alan  Alan  Peter  Joey | Alan  Joey  Lee  Joe  Peter |  | 7  Lyle  Bruce  Alice  Easton  Shawn  Alice  Shawn | Easton  Lyle  Alice  Bruce  Shawn |  | 6  Adam  Adam  Adam Adam  Adam  Adam | Adam |

## Parking Lot

Write a program that:

* Records a **car number** for every car that enters the **parking lot**
* Removes a **car number** when the car leaves the **parking lot**

On the first line, you will receive the number of commands - **N**. On the following **N** lines, you will receive the direction and car's number in the format: **"{direction}, {car\_number}"**. The direction could only be **"IN"** or **"OUT"**. Print the car numbers which are still in the parking lot. Keep in mind that **all car numbers** must be **unique**. If the parking lot is empty, print **"Parking Lot is Empty"**.

***Note:*** The **order** in which we **print** the result does not matter.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10  IN, CA2844AA  IN, CA1234TA  OUT, CA2844AA  IN, CA9999TT  IN, CA2866HI  OUT, CA1234TA  IN, CA2844AA  OUT, CA2866HI  IN, CA9876HH  IN, CA2822UU | CA2844AA  CA9999TT  CA2822UU  CA9876HH |
| 4  IN, CA2844AA  IN, CA1234TA  OUT, CA2844AA  OUT, CA1234TA | Parking Lot is Empty |

## SoftUni Party

There is a party at SoftUni. Many guests are invited, and there are two types of them: **Regular** and **VIP**. When a guest comes, check if they exist on any of the two reservation lists.

On the **first line,** you will receive the number of guests – **N**. On the following **N** lines, you will be receiving their reservation codes. All reservation codes are **8 characters long**, and all **VIP** numbers will start with a **digit**.Keep in mind that **all reservation numbers** must be **unique**.

After that, you will be receiving **guests who came to the party** until you read the **"END"** command.

In the end, print the **number of guests who** **did not come** to the party and **their reservation numbers**:

* **The VIP guests must be first.**
* Both the **VIP** and the **Regular** guests must be **sorted** in **ascending** order.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 5  7IK9Yo0h  9NoBUajQ  Ce8vwPmE  SVQXQCbc  tSzE5t0p  9NoBUajQ  Ce8vwPmE  SVQXQCbc  END | 2  7IK9Yo0h  tSzE5t0p | 6  m8rfQBvl  fc1oZCE0  UgffRkOn  7ugX7bm0  9CQBGUeJ  2FQZT3uC  2FQZT3uC  9CQBGUeJ  fc1oZCE0  END | 3  7ugX7bm0  UgffRkOn  m8rfQBvl |

## Summation Pairs

On the first line, you will receive a sequence of **numbers** separated by space. On the second line, you'll receive a **target** number. Your task is to **find** the **pairs of numbers** whose **sum** **equals** the **target number**. For each found pair print **"{number} + {number} = {target\_number}"**. You may **NOT** use the **same element twice**.

Can you come up with an algorithm that has less time complexity?

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 5 4 2 2 3 1 3 2  4 | 1 + 3 = 4  1 + 3 = 4  2 + 2 = 4 |
| 11 8 5 6 9 2 9 7 3 4  11 | 8 + 3 = 11  5 + 6 = 11  9 + 2 = 11  7 + 4 = 11 |