# Map and Set – Lab

This document defines the exercises for the ["C++ Advanced" course @ Software University](https://softuni.bg/trainings/3658/cpp-advanced-may-2022). Please submit your solutions (source code) to all below-described problems in [Judge](https://judge.softuni.org/Contests/3025/CPlusPlus-Maps-and-Sets-Lab).

Write C++ code for solving the tasks on the following pages.

Code should compile under the C++03 or the C++11 standard.

## Count Same Values in Array

Write a program that counts in a given array of double values the number of occurrences of each value.

#### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| -2.5 4 3 -2.5 -5.5 4 3 3 -2.5 3 | -2.5 - 3 times  4 - 2 times  3 - 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 - 3 times  4 - 6 times  5 - 4 times  3 - 7 times |

## Average Student Grades

Write a program, which reads a **name** of a student and his/her **grades** and **adds** them to the **student record**, then **prints** **the** students' **names** with their **grades** and their **average grade**.

#### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 7  John 5.20  Maria 5.50  John 3.20  Maria 2.50  Sam 2.00  Maria 3.46  Sam 3.00 | John -> 5.20 3.20 (avg: 4.20)  Maria -> 5.50 2.50 3.46 (avg: 3.82)  Sam -> 2.00 3.00 (avg: 2.50) |
| 4  Vlady 4.50  Peter 3.00  Vlady 5.00  Peter 3.66 | Vlady -> 4.50 5.00 (avg: 4.75)  Peter -> 3.00 3.66 (avg: 3.33) |
| 5  George 6.00  George 5.50  George 6.00  John 4.40  Peter 3.30 | George -> 6.00 5.50 6.00 (avg: 5.83)  John -> 4.40 (avg: 4.40)  Peter -> 3.30 (avg: 3.30) |

## Cities by Continent and Country

Write a program that reads **continents**, **countries,** and their **cities** put them in a **nested Map** and **prints** them.

#### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 9  Europe Bulgaria Sofia  Asia China Beijing  Asia Japan Tokyo  Europe Poland Warsaw  Europe Germany Berlin  Europe Poland Poznan  Europe Bulgaria Plovdiv  Africa Nigeria Abuja  Asia China Shanghai | Europe:  Bulgaria -> Sofia, Plovdiv  Poland -> Warsaw, Poznan  Germany -> Berlin  Asia:  China -> Beijing, Shanghai  Japan -> Tokyo  Africa:  Nigeria -> Abuja |
| 3  Europe Germany Berlin  Europe Bulgaria Varna  Africa Egypt Cairo | Europe:  Germany -> Berlin  Bulgaria -> Varna  Africa:  Egypt -> Cairo |
| 8  Africa Somalia Mogadishu  Asia India Mumbai  Asia India Delhi  Europe France Paris  Asia India Nagpur  Europe Germany Hamburg  Europe Poland Gdansk  Europe Germany Danzig | Africa:  Somalia -> Mogadishu  Asia:  India -> Mumbai, Delhi, Nagpur  Europe:  France -> Paris  Germany -> Hamburg, Danzig  Poland -> Gdansk |

## Sets

## Record Unique Names

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

#### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 8  Ivan  Pesho  Ivan  Stamat  Pesho  Alice  Peter  Pesho | Ivan  Pesho  Stamat  Alice  Peter |  | 7  Lyle  Bruce  Alice  Easton  Shawn  Alice  Shawn  Peter | Lyle  Bruce  Alice  Easton  Shawn |  | 6  Roki  Roki  Roki Roki  Roki  Roki | Roki |

## 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.**

The input will be a string in the format: **"direction, carNumber"**. You will be receiving commands until the **"END"** commandisgiven.

Print the car numbers of the cars, which are still in the parking lot:

#### Examples

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

## Unique Usernames

Write a program that reads from the console a sequence of **N usernames** and keeps a collection only of the **unique** ones. On the **first** line, you will be given an integer **N**. On the next **N** lines, you will receive **one** username **per** **line**. Print the collection on the console in **order** of **insertion**:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 6  Ivan  Ivan  Ivan  Pesho  Ivan  NiceGuy1234 | Ivan  Pesho  NiceGuy1234 |

## Sets of Elements

Write a program that prints a **set of elements**. On the first line, you will receive two numbers - **n** and **m**, which represent the lengths of two separate sets. On the next **n** + **m** lines you will receive **n** numbers, which are the numbers in the **first** set, and **m** numbers, which are in the **second** set. Find all the **unique** **elements** that appear in **both of them** and **print** them in the order in which they appear in the **first** set - **n**.

**For example:**

Set with length n = 4: {1, **3**, **5**, 7}

Set with length m = 3: {**3**, 4, **5**}

Set that contains all the **elements** that repeat in **both** **sets** -> {**3**, **5**}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 3  1  3  5  7  3  4  5 | 3 5 |
| 2 2  1  3  1  5 | 1 |

## Periodic Table

Write a program that keeps all the **unique** chemical **elements**. On the first line, you will be given a number **n** - the **count** of input **lines** that you are going to receive. On the next **n** lines, you will be receiving **chemical** **compounds**, separated by a **single** **space**. Your task is to print all the **unique ones** in **ascending** **order**:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Ce O  Mo O Ce  Ee  Mo | Ce Ee Mo O |
| 3  Ge Ch O Ne  Nb Mo Tc  O Ne | Ch Ge Mo Nb Ne O Tc |

## Even Times

Write a program that **prints** a **number** from a collection, which appears an **even** **number** of **times** in it. On the first line, you will be given **n** – the **count** of **integers** you will receive. On the next n lines, you will be receiving **the numbers**. It is **guaranteed** that **only** **one** of them **appears** an **even** **number** of times. Your task is to **find** that **number** and **print** it in the end.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2  -1  2 | 2 |
| 5  1  2  3  1  5 | 1 |