# Lab: Dictionaries

Problems for exercises and homework for the [“Programming Fundamentals Extended” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

## Odd Occurrences

Write a program that extracts from a given sequence of words all elements that present in it **odd number of times** (case-insensitive).

* Words are given in a single line, space separated.
* Print the result elements in lowercase, in their order of appearance.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Java C# PHP PHP JAVA C java | java, c#, c |
| 3 5 5 hi pi HO Hi 5 ho 3 hi pi | 5, hi |
| a a A SQL xx a xx a A a XX c | a, SQL, xx, c |

### Hints

* Use a **dictionary** (**string** → **int**) to count the occurrences of each word (just like in the previous problem).
* Pass through all **key-value pairs** in the dictionary and append to the results list all **keys** that have **odd value**.
* Print the results list.

## Average Student Grades

Write a program, which reads the **name** of a student and their **grades** and **adds** them to the **student record**, then **prints** **grades** along with their **average grade**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 7  Ivancho 5.20  Mariika 5.50  Ivancho 3.20  Mariika 2.50  Stamat 2.00  Mariika 3.46  Stamat 3.00 | Ivancho -> 5.20 3.20 (avg: 4.20)  Mariika -> 5.50 2.50 3.46 (avg: 3.82)  Stamat -> 2.00 3.00 (avg: 2.50) |
| 4  Vladimir 4.50  Petko 3.00  Vladimir 5.00  Petko 3.66  Ivan 4.33 | Vladimir -> 4.50 5.00 (avg: 4.75)  Petko -> 3.00 3.66 (avg: 3.33) |
| 5  Gosho 6.00  Gosho 5.50  Gosho 6.00  Ivan 4.40  Petko 3.30  Petko 4.50 | Gosho -> 6.00 5.50 6.00 (avg: 5.83)  Ivan -> 4.40 (avg: 4.40)  Petko -> 3.30 (avg: 3.30) |

### Hints

* Use a **dictionary** (**string** → **List<double>**)
* Check if the name exists before adding the grade. If it doesn’t, add it to the dictionary.
* Pass through all **key-value pairs** in the dictionary and print the results. You can use the **.Average()** method to quickly calculate the average value from a list.

## Cities by Continent and Country

Write a program to read **continents**, **countries** and their **cities**, put them in a **nested dictionary** and **print** 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 |

### Hints

* Use a **nested** **dictionary** (**string** → (**Dictionary → List<string>)**)
* Check if the continent exists before adding the country. If it doesn’t, add it to the dictionary.
* Check if the country exists, before adding the city. If it doesn’t, add it to the dictionary.
* Pass through all **key-value pairs** in the dictionary and the values’ key-value pairs and print the results.

## Sum, Min, Max, Average

Write a program to read **n** integers and print their **sum**, **min**, **max**, **first**, **last** and **average** values.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5**  12  20  -5  37  8 | Sum = 72  Min = -5  Max = 37  Average = 14.4 |
| **4**  50  20  25  40 | Sum = 135  Min = 20  Max = 50  Average = 33.75 |

### Hints

* Include the “**System.Linq**” namespace to enable aggregate functions.
* Read the input array **nums[]**.
* Use **nums.Min()**, **nums.Max()**, etc.

## Largest 3 Numbers

Read a **list of real numbers** and **print largest 3 of them**. If less than 3 numbers exit, print all of them.

### Examples

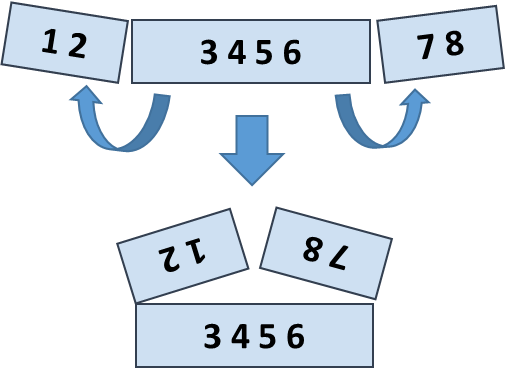
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 10 30 15 20 50 5 | 50 30 20 | 20 30 | 30 20 |

### Hints

You can use LINQ query like this: **nums.OrderByDescending(x** **=>** **x).Take(3)**.

## Fold and Sum

Read an array of **4\*k integers**, **fold** it like shown below, and **print the sum** of the upper and lower rows (**2\*k integers**):



### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5 **2 3** 6 | 7 9 | 5 6 +  2 3 =  7 9 |
| 1 2 **3 4 5 6** 7 8 | 5 5 13 13 | 2 1 8 7 +  3 4 5 6 =  5 5 13 13 |
| 4 3 -1 **2 5 0 1 9 8** 6 7 -2 | 1 8 4 -1 16 14 | -1 3 4 -2 7 6 +  2 5 0 1 9 8 =  1 8 4 -1 16 14 |

**Hints**

Use a **LINQ expression**:

* Row 1, left part: take the **first** **k** numbers and **reverse**.
* Row 1, right part: **reverse** and take the **first** **k** numbers.
* **Concatenate** the **left** and the **right** part of row 1.
* Row 2: skip the **first k** numbers and take the next **2\*k** numbers.
* Sum the arrays **row1** and **row2**: **var** **sum** **=** **row1.Select((x,** **index)** **=>** **x** **+** **row2[index])**.