# Exercises: Strings, Dictionaries, Lambda and LINQ

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

Check your solutions here: <https://judge.softuni.bg/Contests/174/Strings-Dictionaries-Lambda-and-LINQ-Lab>.

## Print String Letters

Read a string and **print its letters** as in the examples below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| SoftUni | str[0] -> 'S'  str[1] -> 'o'  str[2] -> 'f'  str[3] -> 't'  str[4] -> 'U'  str[5] -> 'n'  str[6] -> 'i' |

### Hints

Use a for loop i = 0 … str.Length-1 and print str[i].

## Count Letters in String

**Read a string** and **count how many times each character occurs**. **Print all chars** (case insensitive) **alphabetically** with their counts.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| Alabala | a -> 4  b -> 1  l -> 2 | ooooo, kef | -> 1  , -> 1  e -> 1  f -> 1  k -> 1  o -> 5 | C# Basics | -> 1  # -> 1  a -> 1  b -> 1  c -> 2  i -> 1  s -> 2 |

### Hints

* Allocate array counts[] to hold for each letter its number of occurrences, e.g. count['a'] holds the number of occurrences of 'a'. The array size could be the largest letter in the text + 1: str.Max()+1.
* Iterate over each letter from the input text and increase count[letter].
* Iterate over the count[] array and print all non-zero items and their index converted to letter.

## Print a Receipt

Read a **sequence of numbers** (space separated numbers on a single line) and **print a receipt** of width 24 chars, formatted like at the examples below. Each number should be printed on a separate line with exactly 2 digits after the decimal point, eventually rounded.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 12.5 7 0.50234 | /----------------------\  | 12.50 |  | 7.00 |  | 0.50 |  |----------------------|  | Total: 20.00 |  \----------------------/ | 460 000230 450.6666666 | /----------------------\  | 460.00 |  | 230.00 |  | 450.67 |  |----------------------|  | Total: 1140.67 |  \----------------------/ |

### Hints

You can print the middle receipt row like this: Console.WriteLine("| {0,20:f2} |", num);

## Occurrences in String

Read two strings text and substr and count **how many times** substr **occurs** in the text **as substring**, **case-insensitive**. The input holds two text lines: the **text** and the **substring**. The output consists of an integer number.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| Alabala  la | 2 | aaaabaaa  aa | 5 | huhuhui  huhhu | 0 | Hello, hello  he | 2 |

### Hints

* Use text.IndexOf(word, offset) in a loop.
* Initially use offset = 0.
* After each occurrence is found, search again using offset = occurenceIndex + 1.
* Finish when IndexOf(…) returns -1.
* Think how to perform **case-insensitive matching**.

## Forbidden Substrings

Read a **text** and **several forbidden words**. **Replace** all forbidden words with stars (e.g. **beer** -> **\*\*\*\***). Use "**substring**" **matching** (match part of word), **case-sensitive**. Use the same number of stars like the word length.

### Examples

|  |  |
| --- | --- |
| **Input** | L**earn** **how** to **earn** money or read the HOWto e-l**earn**ing.  beer how programming PHP MySQL earn bitcoins |
| **Output** | L\*\*\*\* \*\*\* to \*\*\*\* money or read the HOWto e-l\*\*\*\*ing |

### Hints

Use string.Replace(word, '\*…\*') in a loop for each word from the input.

## Count Real Numbers

Read a **list of real numbers** and **print them in ascending order** along with their **number of occurrences**.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 8 2.5 2.5 8 2.5 | 2.5 -> 3 times  8 -> 2 times | 1.5 5 1.5 3 | 1.5 -> 2 times  3 -> 1 times  5 -> 1 times | -2 0.33 0.33 2 | -2 -> 1 times  0.33 -> 2 times  2 -> 1 times |

### Hints

* Use SortedDictionary<double, int> named counts.
* Pass through each input number num and increase counts[num] (when num exists in the dictionary) or assign counts[num] = 1 (when num does not exist in the dictionary).
* Pass through all numbers num in the dictionary (counts.Keys) and print the number num and its count of occurrences counts[num].

## 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.

## 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).

## Short Words Sorted

Read a **text**, extract its **words**, find all **short words** (less than 5 characters) and print them **alphabetically**, in **lowercase**.

* Use the following separators: . , : ; ( ) [ ] " ' \ / ! ? *(space)*.
* Use case-insensitive matching.
* Remove duplicated words.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| In SoftUni you can study Java, C#, PHP and JavaScript. JAVA and c# developers graduate in 2-3 years. Go in! | 2-3, and, c#, can, go, in, java, php, you |

### Hints

* To extract the words from the input text, **split** by the specified separators.
* Use a **LINQ expression**:
  + Filter by word length: Where(…)
  + Order by word: OrderBy(…)
  + Use **distinct** to avoid repeated words: Distinct().

## 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]).