# Lab: C# Overview

Problems for exercises and homework for the <https://softuni.bg/trainings/1584/qa-automation-march-2017>.

You can check your solutions here: <https://judge.softuni.bg/Contests/522/CSharp-Overview>.

# Integer and Real Numbers

## Centuries to Minutes

Write program to enter an integer number of **centuries** and convert it to **years**, **days**, **hours** and **minutes**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | 1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes |
| 5 | 5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes |

### Hints

* Use appropriate data type to fit the result after each data conversion.
* Assume that a year has 365.2422 days at average ([the Tropical year](https://en.wikipedia.org/wiki/Tropical_year)).

### Solution

You might help yourself with the code below:



## Circle Area (12 Digits Precision)

Write program to enter a radius r (real number) and **print the area** of the circle with exactly **12 digits** after the decimal point. Use data type of **enough precision** to hold the results.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2.5 | 19.634954084936 |  | 1.2 | 4.523893421169 |

### Hints

* You might use the data type double. It has precision of 15-16 digits.
* To print the output with exactly 12 digits after the decimal point, you might use the following code:



## Exact Sum of Real Numbers

Write program to enter n numbers and calculate and print their **exact sum** (without rounding).

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 3  1000000000000000000  5  10 | 1000000000000000015 |  | 2  0.00000000003  333333333333.3 | 333333333333.30000000003 |

### Hints

* If you use types like float or double, the result will lose some of its precision. Also it might be printed in scientific notation.
* You might use the decimal data type which holds real numbers with high precision with less loss.
* Note that decimal numbers sometimes hold the unneeded zeroes after the decimal point, so 0m is different than 0.0m and 0.00000m.

# Declaring and Invoking Methods

## Blank Receipt

Create a method that prints a blank cash receipt. The method should invoke three other methods: one for printing the header, one for the body and one for the footer of the receipt.

|  |  |
| --- | --- |
| The header should contain the following text: | CASH RECEIPT  ------------------------------ |
| The body should contain the following text: | Charged to\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Received by\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| And the text for the footer: | ------------------------------  © SoftUni |

### Examples

|  |
| --- |
| **Output** |
| CASH RECEIPT  ------------------------------  Charged to\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Received by\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ------------------------------  © SoftUni |

### Hints

1. First create a method with no parameters for printing the header starting with **static void**. Give it a **meaningful name** like "PrintReceiptHeader" and write the code that it will execute:



1. Do the same for printing the receipt body and footer.
2. Create a **method that will call all three methods** in the necessary order. Again, give it a **meaningful and descriptive name** like "PrintReceipt" and write the code:



1. For printing **"©"** use Unicode **"\u00A9"**
2. **Call** (invoke) the PrintReceipt method from the main.



## Sign of Integer Number

Create a method that prints the sign of an integer number n.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | The number 2 is positive. |
| -5 | The number -5 is negative. |
| 0 | The number 0 is zero. |

### Hints

1. Create a method with a **descriptive name** like "PrintSign" which should receive **one parameter** of type **int**.



1. Implement the body of the method by handling different cases:
   1. If the number is greater than zero
   2. If the number is less than zero
   3. And if the number is equal to zero
2. Call (invoke) the newly created method from the main.



## Printing Triangle

Create a method for printing triangles as shown below:

### Examples

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

### Hints

1. After you read the input
2. Start by creating a method **for printing a single line** from a **given start** to a **given end**. Choose a **meaningful name** for it, describing its purpose:



1. Think how you can use it to solve the problem
2. After you spent some time thinking, you should have come to the conclusion that you will need two loops
3. In the first loop you can print the first half of the triangle without the middle line:



1. Next, print the middle line:



1. Lastly, print the rest of the triangle:



## Draw a Filled Square

Draw at the console a filled square of size n like in the example:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 | --------  -\/\/\/-  -\/\/\/-  -------- |

### Hints

1. Read the input
2. Create a method which will print the top and the bottom rows (they are the same). Don’t forget to give it a descriptive name and to give it as a parameter some length
   1. Instead of loop you can use the "new string" command which creates a new string consisting of a character repeated some given times:



1. Create the method which will print the middle rows. Well, of course, you should probably name it "PrintMiddleRow"



1. Use the methods that you've just created to draw a square



# Arrays, Lists and Dictionary

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

You can check your solutions here: <https://judge.softuni.bg/Contests/172/Arrays-Lab>.

## Day of Week

Enter a **day number** [1…7] and print the **day name** (in English) or “**Invalid Day!**”. Use an **array of strings**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | Monday |
| 2 | Tuesday |
| 7 | Sunday |
| 0 | Invalid Day! |

### Hints

* Use an **array of strings** holding the day names: {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}.
* Print the element at index (**day-1**) when it is in the range [1…7] or “**Invalid Day!**” otherwise.

## Reverse an Array of Integers

Write a program to read **an array of integers**, **reverse** it and **print** its elements. The input consists of a **number** n (the number of elements) + n integers, each as a separate line. Print the output on a single line (space separated).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  10  20  30 | 30 20 10 |
| **4**  -1  20  99  5 | 5 99 20 -1 |

### Hints

* First, read the number n.
* Allocate an array of n integers.
* Read the integers in a for-loop.
* Instead of reversing the array, you can just pass through the elements from the last (**n-1**) to the first (**0**) with a reverse for-loop.

## Remove Negatives and Reverse

Read a **list of integers**, **remove all negative numbers** from it and print the remaining elements in **reversed order**. In case of no elements left in the list, print “empty”.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10 -5 7 9 -33 50 | 50 9 7 10 |
| 7 -2 -10 1 | 1 7 |
| -1 -2 -3 | empty |

### Hints

* Read a text line from the console, split it by space, parse the obtained items as integers and convert them to list of integers.
* Create a new empty list for the results.
* Scan the input list from the end to the beginning. Check each element and append all non-negative elements to the result list.
* Finally, print the results list (at a single line holding space-separated numbers).

## Sum Adjacent Equal Numbers

Write a program to **sum all adjacent equal numbers** in a list of decimal numbers, starting from **left to right**.

* After two numbers are summed, the obtained result could be equal to some of its neighbors and should be summed as well (see the examples below).
* Always sum the **leftmost** two equal neighbors (if several couples of equal neighbors are available).

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanation** |
| 3 3 6 1 | 12 1 | **3 3** 6 1 🡪 **6 6** 1 🡪 12 1 |
| 8 2 2 4 8 16 | 16 8 16 | 8 **2 2** 4 8 16 🡪 8 **4 4** 8 16 🡪 **8 8** 8 16 🡪 16 8 16 |
| 5 4 2 1 1 4 | 5 8 4 | 5 4 2 **1 1** 4 🡪 5 4 **2 2** 4 🡪 5 **4 4** 4 🡪 5 8 4 |

### Hints

1. Read the **input** and parse it to **list of numbers**.
2. Find the **leftmost** two **adjacent equal cells**.
3. **Replace** them with their **sum**.
4. **Repeat** (1) and (2) until no two equal adjacent cells survive.
5. **Print** the processed list of numbers.

## Split by Word Casing

Read a **text**, split it into words and distribute them into **3 lists**.

* **Lower-case words** like “programming”, “at” and “databases” – consist of lowercase letters only.
* **Upper-case words** like “PHP”, “JS” and “SQL” – consist of uppercase letters only.
* **Mixed-case words** like “C#”, “SoftUni” and “Java” – all others.

Use the following **separators** between the words: , ; : . ! ( ) " ' \ / [ ] space

Print the 3 lists as shown in the example below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Learn programming at SoftUni: Java, PHP, JS, HTML 5, CSS, Web, C#, SQL, databases, AJAX, etc. | Lower-case: programming, at, databases, etc  Mixed-case: Learn, SoftUni, Java, 5, Web, C#  Upper-case: PHP, JS, HTML, CSS, SQL, AJAX |

### Hints

* **Split** the input text using the above described **separators**.
* **Process** the obtained **list of words** one by one.
* Create 3 lists of words (initially empty): lowercase words, mixed-case words and uppercase words.
* Check each word and append it to one of the above 3 lists:
  + Count the **lowercase letters** and **uppercase letters**.
  + If all letters are **lowercase**, append the word to the lowercase list.
  + If all letters are **uppercase**, append the word to the uppercase list.
  + Otherwise the word is considered mixed-case 🡪 append it to the mixed-case list.
* Print the obtained 3 lists as shown in the example above.

## 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].

# LINQ

## 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).

## 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]).

# Objects and Classes

## Day of Week

You are given a **date** in format **day-month-year**. Calculate and print the **day of week** in **English**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 18-04-2016 | Monday |
| 27-11-1996 | Wednesday |

### Hints

* **Read the date as string** from the Console.
* Use the method [**DateTime.ParseExact(string date, format, provider)**](https://msdn.microsoft.com/en-us/library/w2sa9yss(v=vs.110).aspx) to convert the input string to object of typeDateTime. Use format **“**d-M-yyyy**”** and CultureInfo.InvariantCulture.
  + Alternatively split the input by “-“ and you will get the day, month and year as numbers. Now you can create new DateTime(year, month, day).
* The newly created DateTime object has property [**DayOfWeek**](https://msdn.microsoft.com/en-us/library/system.datetime.dayofweek(v=vs.110).aspx).

## Distance Between Points

Write a method to calculate the distance between two points **p1** {**x1**, **y1**} and **p2** {**x2**, **y2**}. Write a program to read **two points** (given as two integers) and print the **Euclidean distance** between them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 4  6 8 | 5.000 |
| 3 4  5 4 | 2.000 |
| 8 -2  -1 5 | 11.402 |

### Hints

* Create a **class** Point holding properties X and Y.
* Write a method CalcDistance(Point p1, Point p2) that returns the distance between the given points – a double number.
* Use [this formula](http://www.cut-the-knot.org/pythagoras/DistanceFormula.shtml) to calculate the distance between two points. How it works?
  + Let's have two points **p1** {**x1**, **y1**} and **p2** {**x2**, **y2**}
  + Draw a right-angled triangle
  + Side **a = |x1 - x2|**
  + Side **b = |y1 - y2|**
  + Distance == side **c** (hypotenuse)
  + **c2** = **a2** + **b2** (Pythagorean theorem)
  + Distance = **c** =



* You can use [**Math.Sqrt(number)**](https://msdn.microsoft.com/en-us/library/system.math.sqrt(v=vs.90).aspx) method for calculating a square root.

## Closest Two Points

Write a program to read **n** points and find the **closest two** of them.

### Input

The **input** holds the number of points n and n lines, each holding a point {X and Y coordinate}.

### Output

* The **output** holds the shortest distance and the closest two points.
* If several pairs of points are equally close, print **the first** of them (from top to bottom).

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Visualization** | **Comments** |
| 4  3 4  6 8  2 5  -1 3 | 1.414  (3, 4)  (2, 5) |  | The closest two points are **{3, 4}** and **{2, 5}** at distance 1.4142135623731 ≈ **1.414**. |
| 3  12 -30  6 18  6 18 | 0.000  (6, 18)  (6, 18) |  | Two of the points have the same coordinates **{6, 18}**, so the distance between them is **0**. |
| 3  1 1  2 2  3 3 | 1.414  (1, 1)  (2, 2) |  | The pairs of points {{1, 1}, {2, 2}} and {{2,2}, {3,3}} stay at the same distance, but the first pair is {**{1, 1}**, **{2, 2}**}. The distance between them is 1.4142135623731 ≈ **1.414**. |

### Hints

* Use the **class** Point you created in the previous task.
* Create an array Point[]points that will keep all points.
* Create a method Point[]FindClosestPoints(Point[] points) that will check distance **between every two pairs** from the array of points and returns the two closest points in a new array.
* Print the **closest distance** and the **coordinates** of the two closest points.

# String and Text Processing

## Reverse String

Write a program that reads a string from the console, reverses it and prints the result back at the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| sample | elpmas |
| 24tvcoi92 | 29iocvt42 |

## Count Substring Occurrences

Write a program to **find how many times a given string appears in a given text as substring**. The text is given at the first input line. The search string is given at the second input line. The output is an integer number. Please ignore the **character casing**. **Overlapping** between occurrences is **allowed**. Examples:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Wel**come to the Software University (SoftUni)! **Wel**come to programming. Programming is **wel**lness for developers, said Max**wel**l.  wel | 4 |
| **aaaaaa**  aa | 5 |
| **ababa** c**aba**  aba | 3 |
| Welcome to SoftUni  Java | 0 |

## Text Filter

Write a program that takes a **text** and a **string of banned words**. All words included in the ban list should be replaced with **asterisks** "**\***", equal to the word's length. The entries in the ban list will be separated by a **comma** and **space** "**,** ".

The ban list should be entered on the first input line and the text on the second input line. Example:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Linux, Windows  It is not **Linux**, it is GNU/**Linux**. **Linux** is merely the kernel, while GNU adds the functionality. Therefore we owe it to them by calling the OS GNU/**Linux**! Sincerely, a **Windows** client | It is not \*\*\*\*\*, it is GNU/\*\*\*\*\*. \*\*\*\*\* is merely the kernel, while GNU adds the functionality. Therefore we owe it to them by calling the OS GNU/\*\*\*\*\*! Sincerely, a \*\*\*\*\*\*\* client |

## Palindromes

Write a program that extracts from a given text all palindromes, e.g. ABBA, lamal, exe and prints them on the console on a single line, separated by comma and space. Use spaces, commas, dots, question marks and exclamation marks as word delimiters. Print only **unique** palindromes, **sorted** lexicographically.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hi,exe? ABBA! Hog fully a string. Bob | a, ABBA, exe |