# Lab: JS Fundamentals

Lab problems for the ["Back-End Technologies Basics"](https://softuni.bg/trainings/4398/back-end-technologies-basics-january-2024) Course @ SoftUni.   
You can check your solutions in [Judge](https://judge.softuni.org/Contests/4640/JS-Fundamentals-Lab).

## Sum First and Last Array Elements

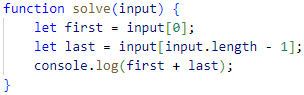
Write a function that receives an **array of numbers** and prints the sum of the **first** and **last** element in that array.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [20, 30, 40] | 60 |
| [10, 17, 22, 33] | 43 |
| [11, 58, 69] | 80 |

### Hints

You can access the **last element** in an array by subtracting **1** from **its length**:



## Reverse an Array of Numbers

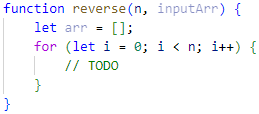
Write a program, which receives a number n and an **array** of elements. Your task is to **create** a new array with n numbers from the original array, **reverse** it and **print** its elements on a single line, space-separated.

### Examples

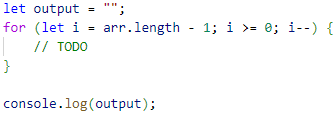
|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, [10, 20, 30, 40, 50] | 30 20 10 |
| 4, [-1, 20, 99, 5] | 5 99 20 -1 |
| 2, [66, 43, 75, 89, 47] | 43 66 |

### Hints

Use push() to add elements inside the new array.



Print the output.



## Even and Odd Subtraction

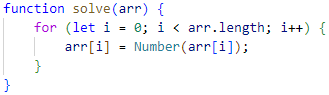
Write a program that calculates the **difference** between the sum of the **even** and the sum of the **odd** numbers in an array.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| [1,2,3,4,5,6] | 3 | 2 + 4 + 6 = 12, 1 + 3 + 5 = 9, 12 - 9 = 3 |
| [3,5,7,9] | -24 |  |
| [2,4,6,8,10] | 30 |  |

### Hints

Parse each string to number.



Create two variables – for **even** and **odd** sum.



Iterate through all elements in the array with aloopand check if the **number** is **odd** or **even**.

Print the difference.

## Substring

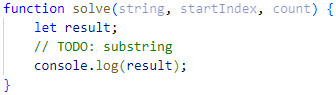
Write a function that **receives a string** and **two numbers**. The numbers will be a **starting index** and **count** of elements to substring. Print the result.

### Input / Output

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'ASentence', 1, 8 | Sentence |
| 'SkipWord', 4, 7 | Word |

### Hints

Create a new string that takes the needed count of elements from the given string.



## Censored Words

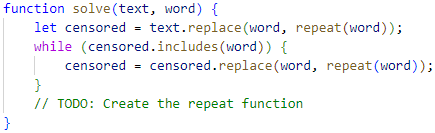
Write a function that **receives a text as** a first parameter and a **single word** as a second. Find **all occurrences** of that word in the text and replace them with the corresponding count of **'\*'**.

### Input / Output

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'A small sentence with some words', 'small' | A \*\*\*\*\* sentence with some words |
| 'Find the hidden word', 'hidden' | Find the \*\*\*\*\*\* word |

### Hints

Save the new text in a new variable.



The repeat() function should take the length of the word and return that count of stars '\*'.

## Count String Occurrences

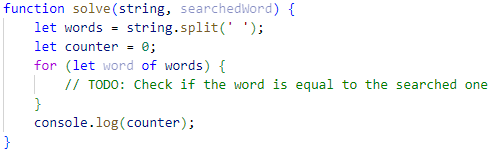
Write a function that **receives a text** and a **single word** **that you need to search**. Print the number of all occurrences of this word in the text.

### Input / Output

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'This is a word and it also is a sentence',  'is' | 2 |
| 'softuni is great place for learning new programming languages',  'softuni' | 1 |

### Hints

Split the sentence into words and create a **counter** that stores how many times the searched word occurs.



## Format Grade

Write a function that **receives a grade** between **2.00** and **6.00** and **prints** a formatted line with **grade and description.**

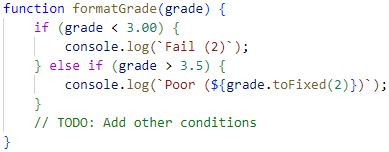
* < 3.00 - "**Fail**"
* >= 3.00 and < 3.50 - "**Poor**"
* >= 3.50 and < 4.50 - "**Good**"
* >= 4.50 and < 5.50 - "**Very** **good**"
* >= 5.50 - "**Excellent**"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3.33 | Poor (3.33) |
| 4.50 | Very good (4.50) |
| 2.99 | Fail (2) |

### Hints

Use a series of if statements checking the threshold between grade brackets



## Math Power

Write a function that **calculates** and **print** the value of a number **raised** to a **given power**:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2,8 | 256 |
| 3,4 | 81 |

### Hints

Create a function that will have **two parameters** - the **number** and the **power**.

**Print** the result to the console.

## Repeat String

Write a function that receives a **string** and a **repeat** **count** n. The function should **return** a new string (the old one repeated **n** times).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| "abc", 3 | abcabcabc |
| "String", 2 | StringString |

### Hints

Use a loop or another method to repeat the input string.  
Use the return operator to produce the result.

## Orders

Write a function that calculates the **total** **price** of an order and prints it on the console. The function should receive one of the following products: **coffee, coke, water, snacks**; and a **quantity** of the product. The **prices** for a single piece of each product are:

* coffee – 1.50
* water – 1.00
* coke – 1.40
* snacks – 2.00

Print the result **formatted** to the **second** **decimal** **place**.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| "water", 5 | 5.00 |
| "coffee", 2 | 3.00 |

### Hints

Create a function and pass the two variables to it.  
Print the result in the function.

### Hints

Use a switch statement for the different operators.

## Person Info

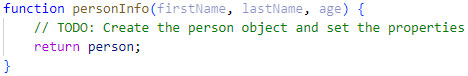
Write a function that receives **3 parameters**, sets them to an **object**, and **returns** that object.

The input comes as **3 separate strings** in the following order: **firstName**, **lastName**, **age**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Object Properties** |
| "Peter",  "Pan", "20" | firstName: Peter  lastName: Pan  age: 20 |
| "George",  "Smith", "18" | firstName: George  lastName: Smith  age: 18 |

### Hints



## City

Write a function that receives a **single** **parameter** – an **object**, containing **five properties**:

**{ name, area, population, country, postcode }**

Loop through all the **keys** and **print** them with their **values** in format: "**{key} -> {value}**"

See the examples below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| {  name: "Sofia",  area: 492,  population: 1238438,  country: "Bulgaria",  postCode: "1000"  } | name -> Sofia  area -> 492  population -> 1238438  country -> Bulgaria  postCode -> 1000 |
| {  name: "Plovdiv",  area: 389,  population: 1162358,  country: "Bulgaria",  postCode: "4000"  } | name -> Plovdiv  area -> 389  population -> 1162358  country -> Bulgaria  postCode -> 4000 |

## Phone Book

Write a function that stores information about a **person's name** and **phone number**. The input is an **array of strings** with space-separated name and number. **Replace duplicate names**. Print the result as shown.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Tim 0834212554',  'Peter 0877547887',  'Bill 0896543112',  'Tim 0876566344'] | Tim -> 0876566344  Peter -> 0877547887  Bill -> 0896543112 |
| ['George 0552554',  'Peter 087587',  'George 0453112',  'Bill 0845344'] | George -> 0453112  Peter -> 087587  Bill -> 0845344 |