# Exercises: Functions and Logic Flow

Problems for exercise and homework for the ["JavaScript Fundamentals Course@SoftUni".](https://softuni.bg/trainings/2247/js-fundamentals-january-2019) Submit your solutions in the SoftUni Judge System at <https://judge.softuni.bg/Contests/1450>

## Leap Year

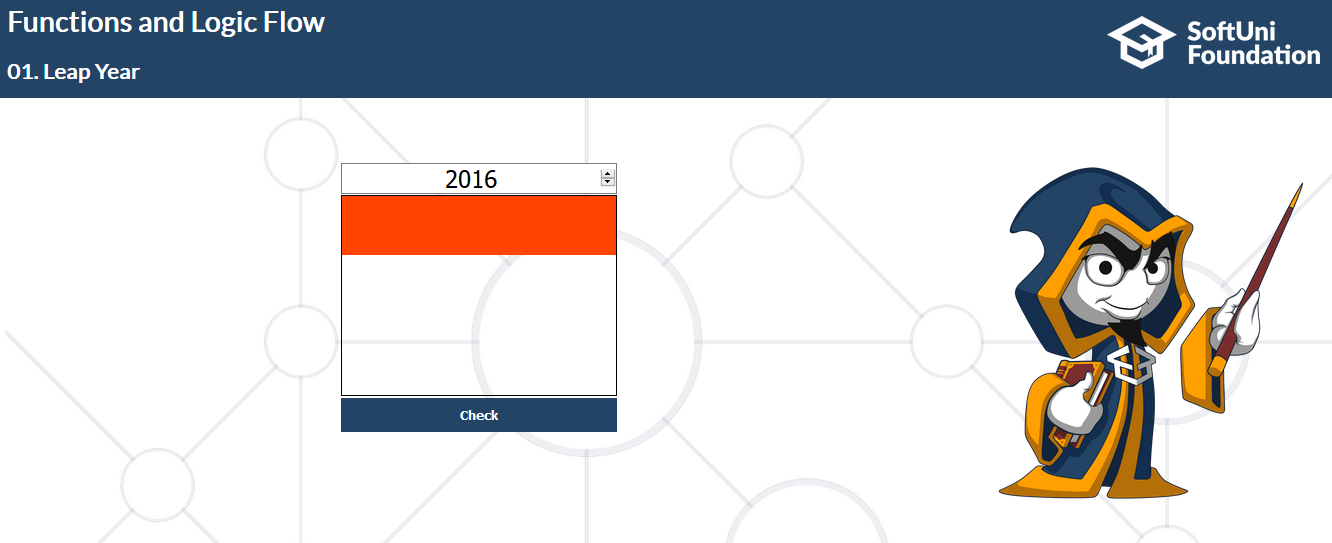
You are given a **year** as an **input**. Your task is to find if the given year is **leap** or **not**.

When the “**Check**” **button** is **clicked** the **h2 element** inside the div with **id** “**yea**r” should be filled with “**Leap Year**” or “**Not Leap Year**” **depends** on the result. And the **div element** inside the div with **id** ”**year**” should be filled with the **given year**.

After every click on the “Check” button the input field **must be cleared**!

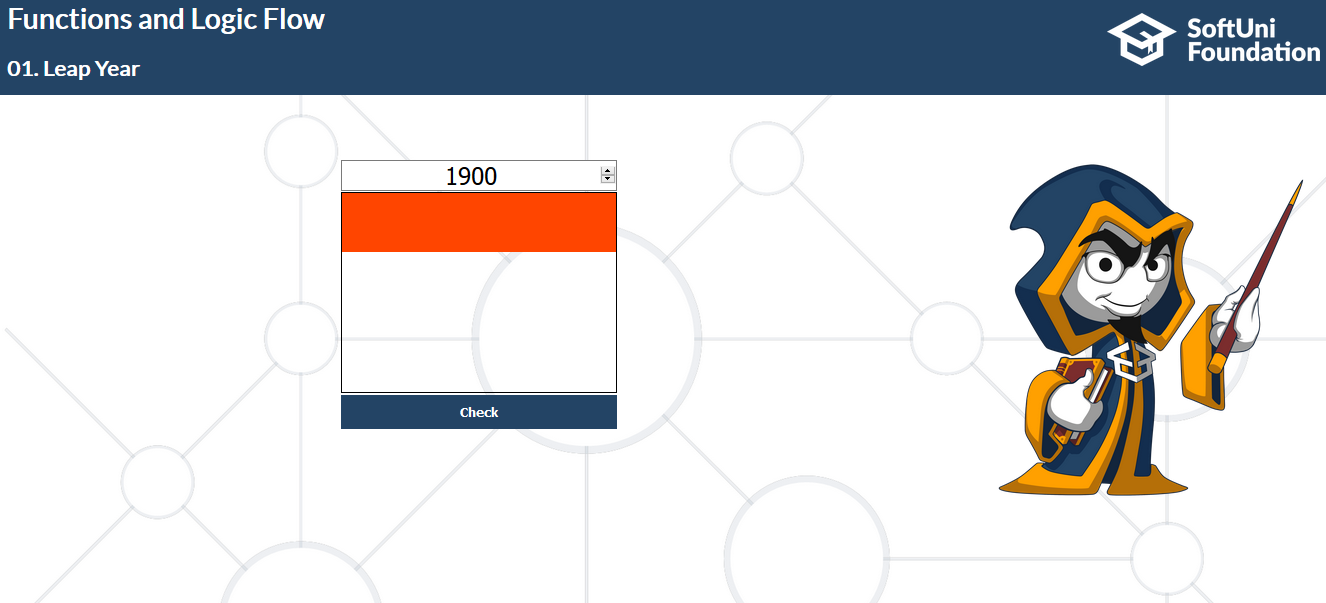
### Examples

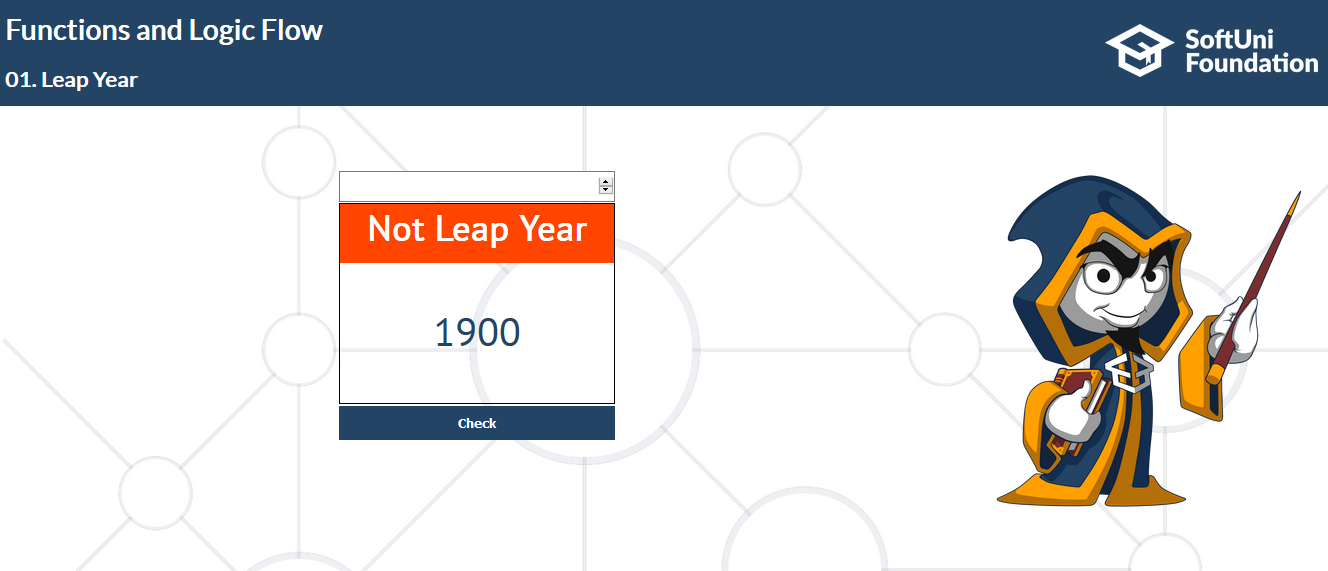
Input: **2016**





Input: **1900**





## Simple Number Validator

Your task here is to create a **Simple Number Validator**. This Validator calculates whether the given input of **digits** form a **valid** number. The given number is considered to be valid:

If the **last digit** (checkNum) is **equal** to the **sum** (sumNums) of the product of the first nine digits with their **weight** (weightPosition), which is a constant for each number and is equal to the position it takes. **SumNums** is the value of the remainder of the division between **weightSum** and the **number 11**.  
Have in mind that if the remainder is 10, you have to keep 0 as a value.

As **input** you will receive a **10-digit number**

You are also given an array of 9 numbers (**weightPosition**), as each number corresponds to a given position of **numsRow** => **let weightPosition = [2, 4, 8, 5, 10, 9, 7, 3, 6];**

### Constrains

The row of numbers you receive will be a string and every digit will have a value between **0** and **9**.

The output should be true (if the row is valid) or false (if the row is invalid).

### Output

When you check if the given number is **valid or not**. Print one of the following messages depends on the result:

**This number is Valid!**

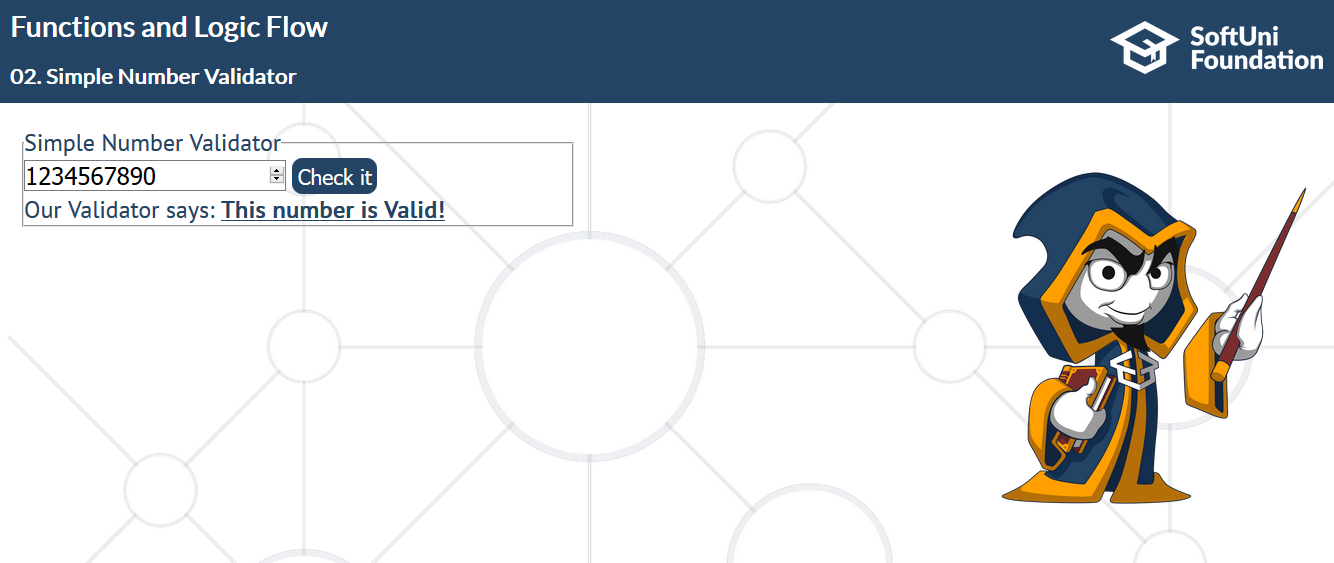
**This number is NOT Valid!**

This message must appear like **text** into **span element** with **id** ‘**response’**;

### Examples

Input: **1234567890**





Input: **7890123456**





## Date Validator

Write a JS program that checks whether a **given date is valid**. You will receive three parameters - **year**, **month** and **day**. When “**Check this date**” button is clicked you need to check the date, create a **new paragraph** and **fill it** with the **result** of the check for the current date. This paragraph is expected to be in the **div element** with **id** “**output**”;

Print “**Data {year}-{month}-{day} is valid**” for **valid** date  
and  
“**Data {year}-{month}-{day} is not valid**” for **invalid** date.

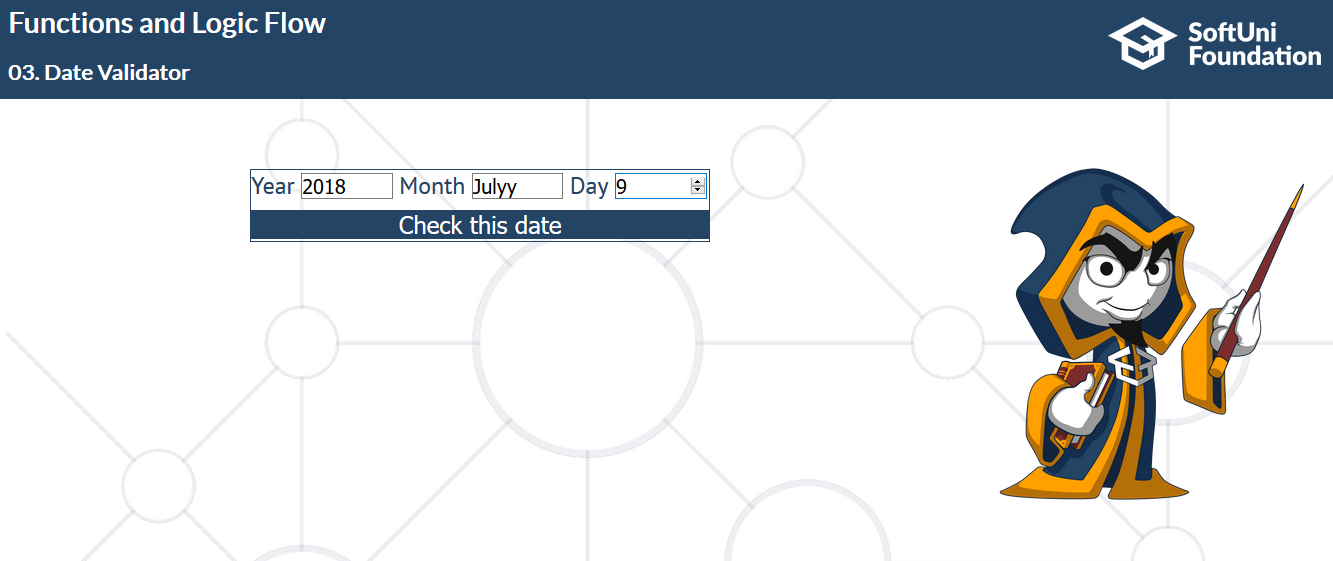
### Constrains

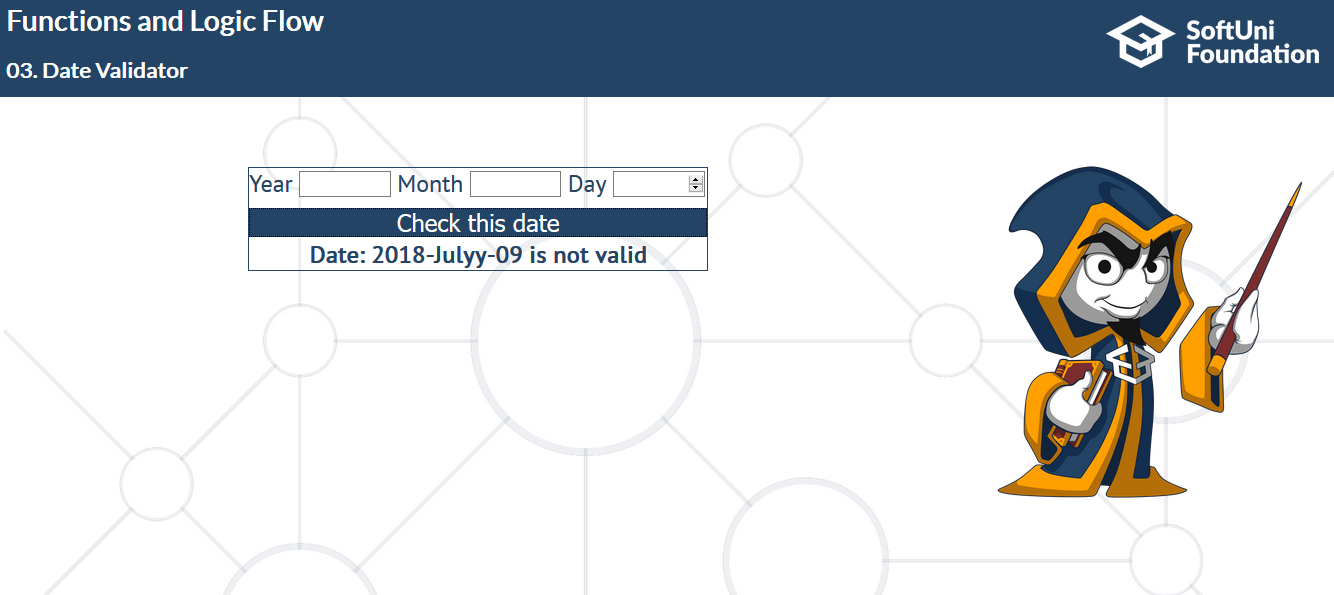
**Year** needs to be **numbe**r between **1** and **9999**  
**Month** needs to be **1 Word Capitalized String** – January, February, March (**v alid month**);  
**Day** needs to be **number > 0 and < 32 depends on the month;**

**The three fields MUST form a VALID DATE.**

### Examples

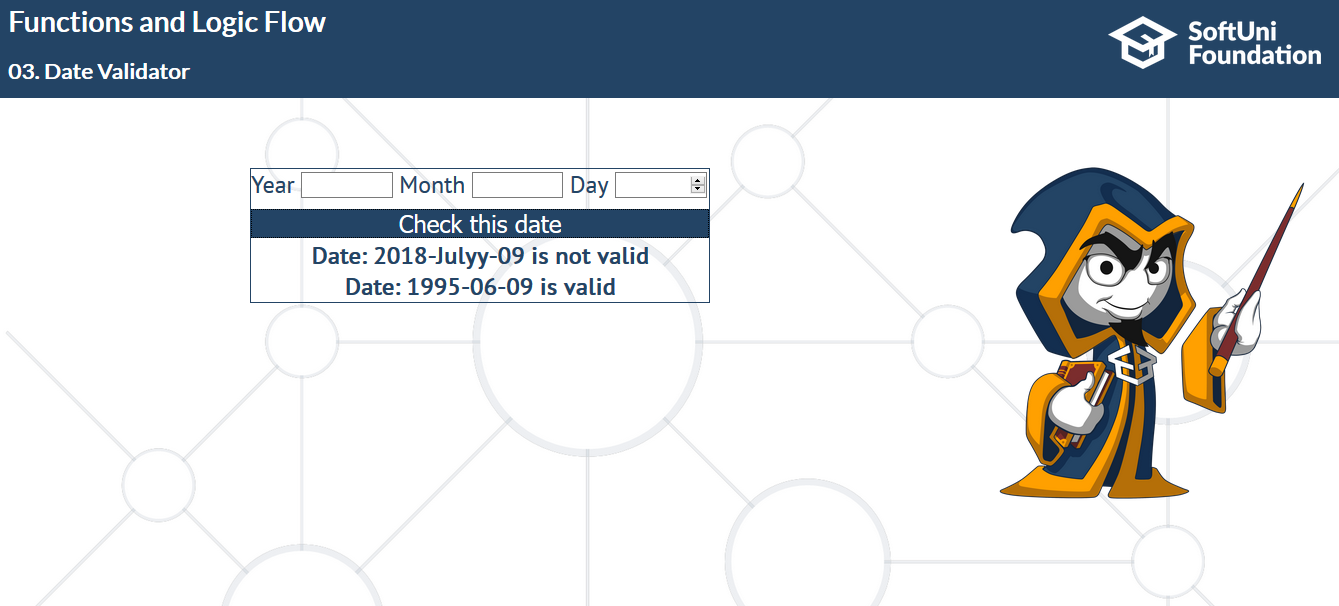
Input for Year: **2018**, Input for Month: **Julyy**, Input for Day: **9**





Input for Year: **1995**, Input for Month: **July,** Input for Day: **9**





## EGN Generator

Your next task is to create an **EGN generator**. **EGN** consists of 10 digits from 0 to 9, ordered in the following sequence:

**The first two digits** are the **last two** from the **year of birth**.

**The next two digits** are **the month**.

**The next two** are **the date**.

**The next three digits** are for **the region** in which **the person was born**, **as the last one** is for **the gender**.   
Even numbers are used for males and, therefore, odd ones for females.

**The last digit** **validates** **the EGN**. It is formed by summing the products of all 9 digits (**weightSum**) with their weight (**weightPosition**) which is a constant for each digit and equals to the position it takes. **CheckNum** is the value of the remainder of the division between **weightSum** and the number 11.  
**Have in mind that if the remainder is 10, you have to keep 0 as a value.**

**weightPosition = [2, 4, 8, 5, 10, 9, 7, 3, 6];**

### Input

You will receive **five parameters**, as follows:

**Year** (number), **Month** (string), **Date**(day) (number), **gender** (string), **regional** **code** (number);

### Output

When “**GET MY EGN**” button is clicked you have to **generate** a **new EGN** with the given information and put it into **paragraph** **element** with **id** "**egn**".

### Constrains

**After each click of the button, input fields must be reset in their original state**.  
**Valid year** is between **1900** and **2100** (**including**);

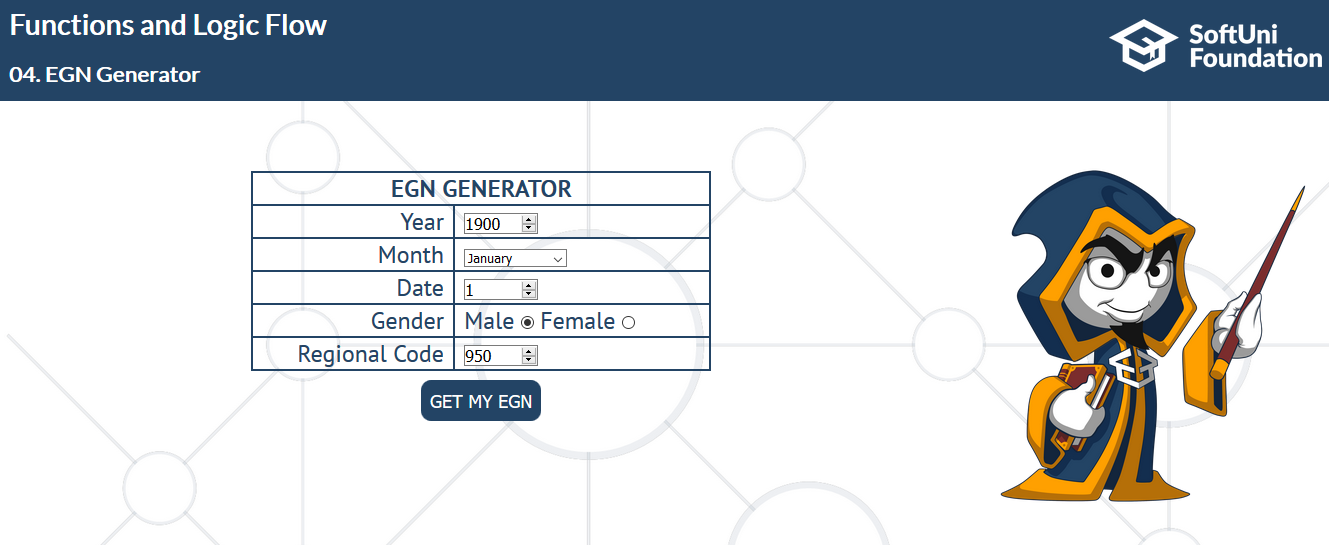
**Valid regional code** is between **43** and **999** (**including**);

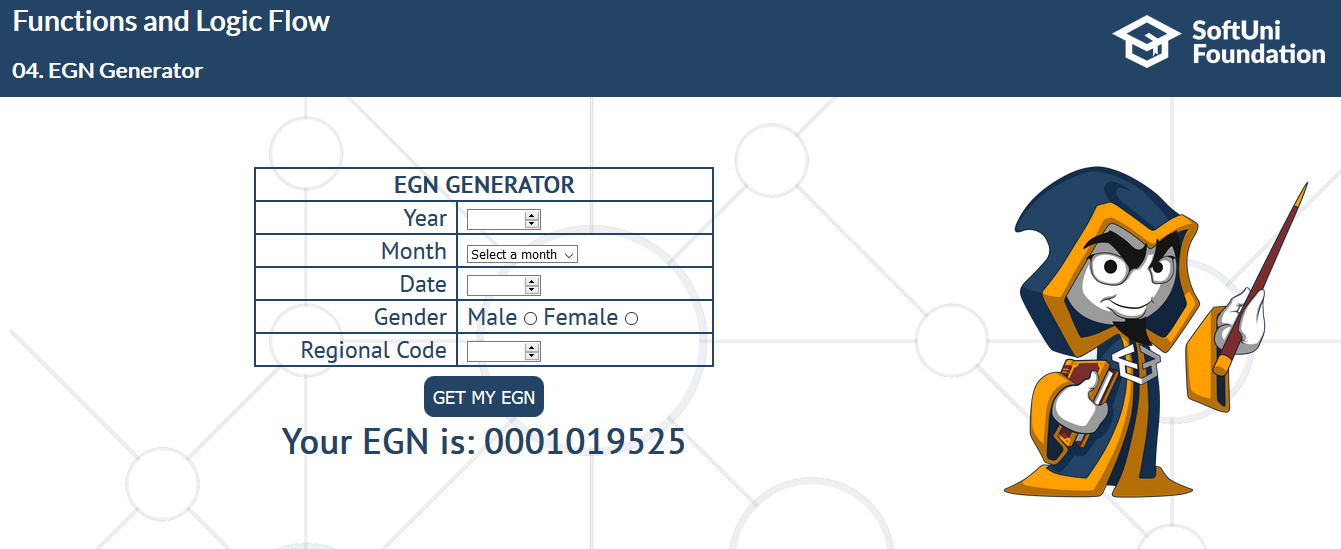
**List of area codes:**

Blagoevgrad - 43  
Burgas 43 – 93  
Varna 93 -139  
Veliko Turnovo 139- 169  
Vidin 183  
Vratca 183- 217  
Gabrovo 217- 233  
Kurdjali- 233 -281  
Kiustendil 281- 301  
Lovech 301- 319  
Montana 319 -341  
Pazardjik 341- 377  
Pernik 377- 395  
Pleven 395- 435  
Plovdiv 435- 501  
Razgrad 501- 527  
Ruse 527- 555  
Silistra 555 -575  
Sliven 575 -601  
Smolqn 601- 623  
Sofia – city 623- 721  
Sofia – region 721- 751  
Stara Zagora 751- 789  
Dobrich (Tolbuhin) 789- 821  
Turgovishte- 821- 843  
Haskovo 843- 871  
Shumen 871- 903  
Qmbol 903 -925  
Other/Unknown 925 – 999

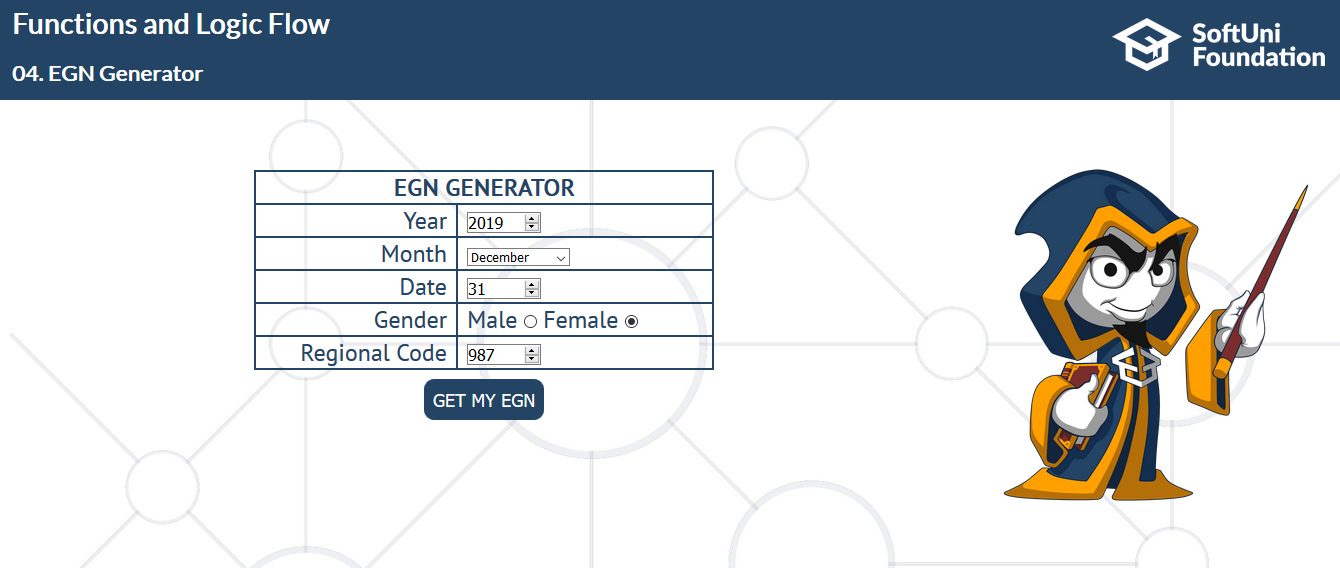
### Examples

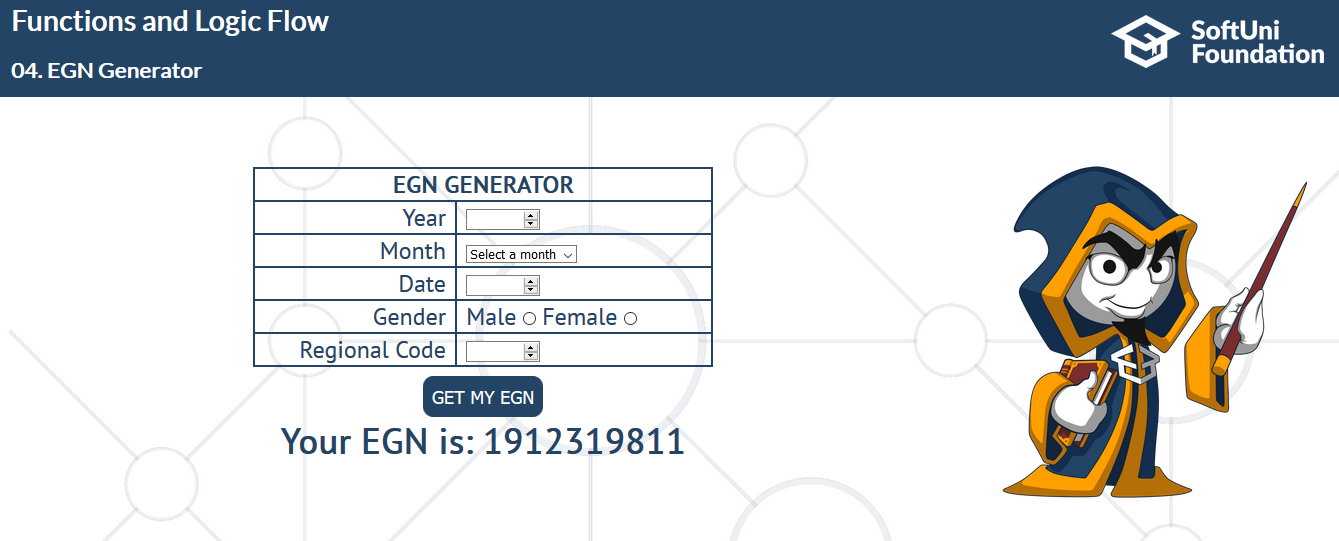
**Year: 1900, Month: January, Date: 1, Gender: Male, Regional Code: 950**





**Year: 2019, Month: December, Date: 31, Gender: Female, Regional Code: 987**





## Cooking Numbers

Write a JS program that receives a number and a list of five operations. Perform the operations in sequence by starting with the input number and using the result of every operation as starting point for the next. Print the result of every operation in order. The operations can be one of the following:

* chop – divide the number by two
* dice – square root of number
* spice – add 1 to number
* bake – multiply number by 3
* fillet – subtract 20% from number

### Input

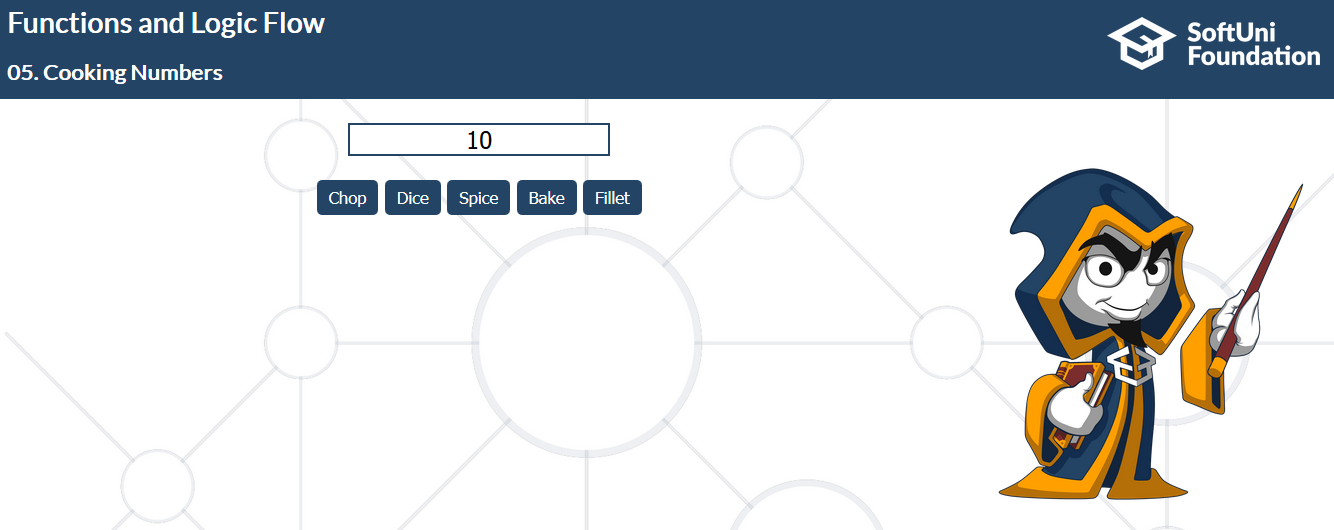
The original (first) numbers comes from **input field.  
If in the input field you do NOT receive any number, you should work with zero (0);**

### Output

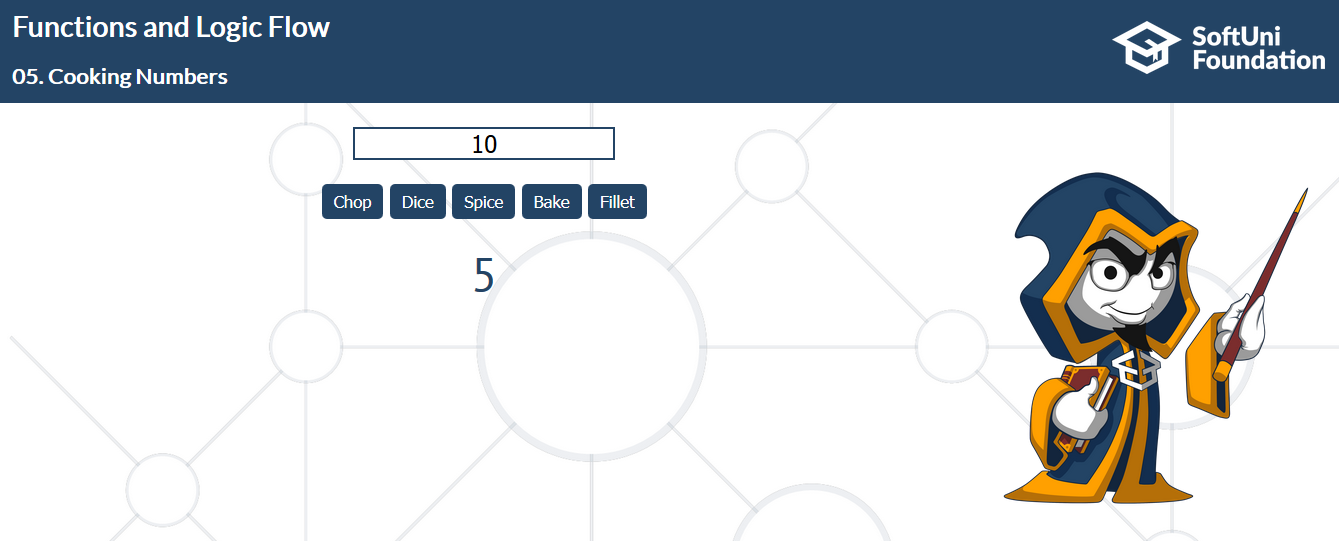
**After every click** on the operation button you should perform the necessary action and print the result into the paragraph with id (output)

### Example

The actions in this example will be in this specific order: **Chop** -> **Dice** -> **Spice** -> **Bake** and **Fillet.**

**The initial number will be 10.**

**Chop**



**Dice**



**Spice**



**Bake**



**Fillet**



## Cards Generator

In this problem you should write a **JS functions** that generates a hand of cards, depends on **starting** and **ending** **card**.

Every card should be **div element with class ‘card’.** Also needs to contain **3 paragraphs**. **The first** and the **last one** have to contain the **Unicode character** of that **suit. The middle** one have to contain the **current card value**. (**2**…**A**)

All cards must be appended to the **section** with **id** “**cards**”.

### Input

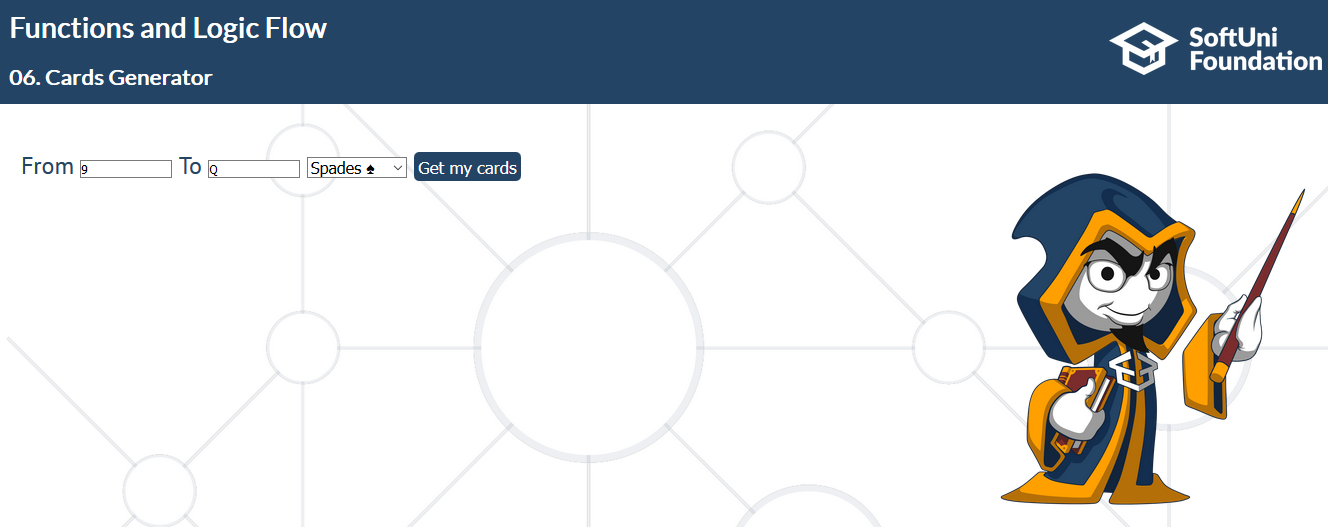
**From** and **To** **values** will be in range: **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, **J**, **Q**, **K** and **A**.

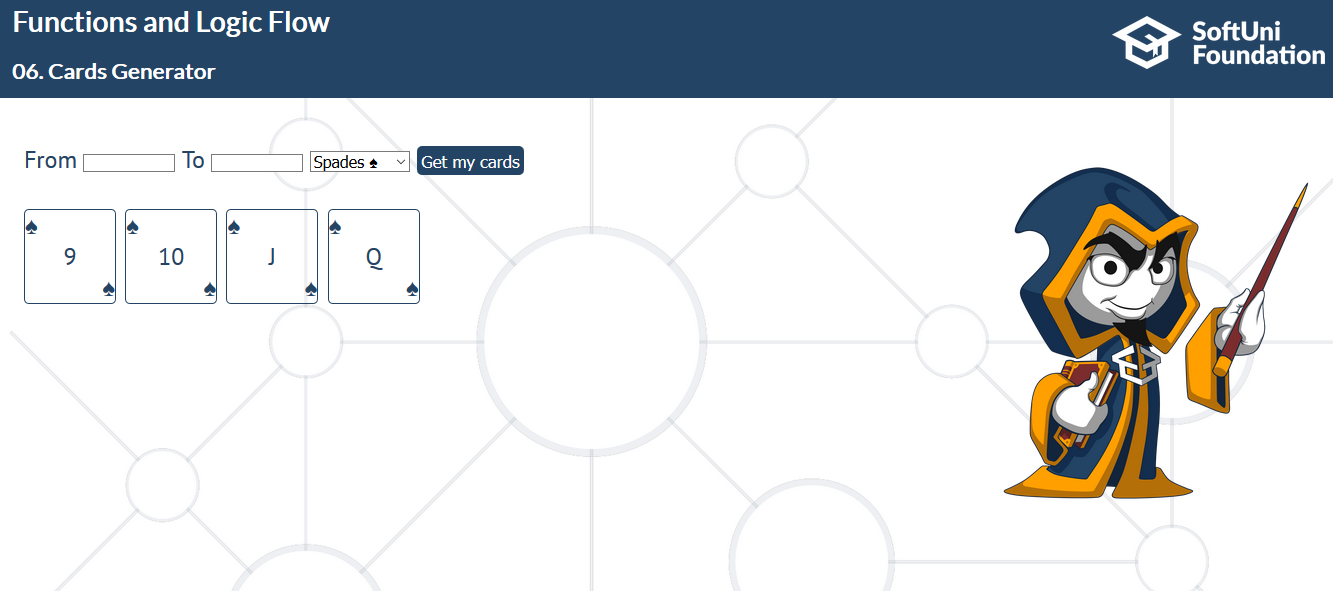
**Suit** will be one of the following: **Hearts**, **Diamonds**, **Spades** or **Clubs;**

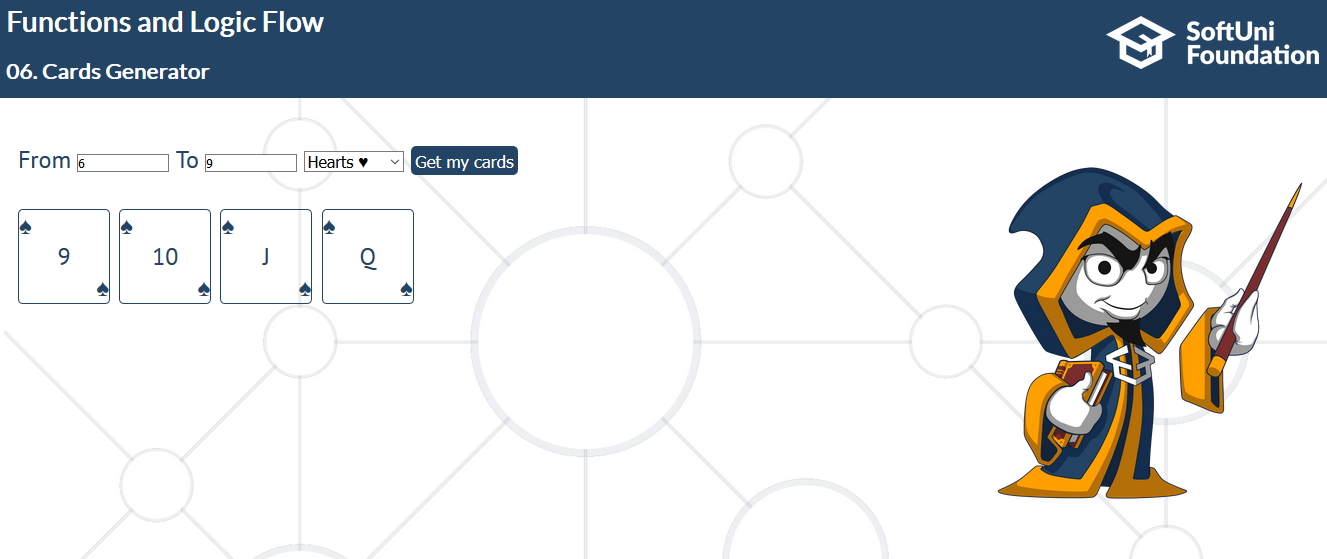
**Note: From card value** will be **greater** or **equal** to **To card value.**

**Note:** If you already have some cards into the **cards section** you need to append the new cards to the old ones.

From: **9**, To: **Q**, **Spades**





From: **6**, To: **9**, **Hearts** 



## Greatest Common Divisor

Write a function that finds the [greatest common divisor](https://en.wikipedia.org/wiki/Greatest_common_divisor) of two numbers.

### Input

The input comes as **two number parameters**.

### Output

Print the result in the following format: **"Greatest Common Divisor: {result}"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2154, 458 | Greatest Common Divisor: 2 |
| 2000, 1000 | Greatest Common Divisor: 1000 |
| 255, 486 | Greatest Common Divisor: 3 |

### Hints



## Binary Search

Write a function that does a [binary search](https://en.wikipedia.org/wiki/Binary_search_algorithm) in an array and prints a result.

### Input

You will be given **two parameters**:

* A **sorted array with numbers**
* A **single number** to search for

### Output

If the number **is present** in the array print: **"Found number {number} at index {index}"**

If the number is not present in the array print: **"The number {number} is not in the array."**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [10, 11, 15, 23, 25, 32], 15 | Found 15 at index 2 |
| [13, 15, 17, 21, 26, 67, 87, 88, 90], 20 | 20 is not in the array |

### Hints



## Hailstone sequence

Write a function that generates the [hailstone sequence](https://rosettacode.org/wiki/Hailstone_sequence) starting from a given number.

### Input

The input comes as a **single number**

### Output

Print the **sequence on a single line separated by space**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 13 | 0 20 10 5 16 8 4 2 1 |
| 3 | 3 10 5 16 8 4 2 1 4 2 1 |

### Hints



## Dot Product \*\*

Write a function that generates the [dot product](https://www.mathsisfun.com/algebra/matrix-multiplying.html) of two matrices.

### Input

The input will come as **two parameters: two matrices**.

### Output

Print each row from the **resulting matrix** in individual **paragraph element** inside a **div** element with **id** “**result**”.  
Look the example below.

### Constrains

You need to write a **transpose** function that transposes **the second matrix** in order for the calculation of the dot product to be possible

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [[1, 2, 3], [4, 5, 6]] ,  [[7, 9, 11], [8, 10, 12]] | 58 64  139 154 |

**Hints**



## Factors

Write a function to compute all of the **factors** of a given number

### Input

A **single number**

### Output

Print the sequence **starting from 1 in the format shown in the example**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 15 | 1 3 5 15 |
| 21 | 1 3 7 21 |

**Hints**

