

# Lab: Strings and Regular Expressions

Problems for in-class lab for the [“JavaScript Fundamentals” course @ SoftUni](#). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/312>.

## 1. Print Letters

Write a JS function that prints all the symbols of a string, each on a new line.

The **input** comes as array of one string element.

The **output** is printed on the console, each letter on a new line.

### Examples

Input	Output
['Hello, World!']	str[0] -> H str[1] -> e str[2] -> l str[3] -> l str[4] -> o str[5] -> , str[6] -> str[7] -> W str[8] -> o str[9] -> r str[10] -> l str[11] -> d str[12] -> !
['SoftUni']	str[0] -> S str[1] -> o str[2] -> f str[3] -> t str[4] -> U str[5] -> n str[6] -> i

## 2. Concatenate Reversed

Write a JS function that reverses a series of strings and prints them concatenated from last to first.

The **input** comes as array of strings.

The **output** is printed on the console. Print all strings concatenated on a single line, starting from the last input string, going to the first. Reverse each individual string's letters.

### Examples

Input	Output
['I', 'am', 'student']	tnedutsmal
['race', 'car']	racecar

### 3. Count Occurrences

Write a JS function that counts how many times a string occurs in a given text. Overlapping strings are allowed.

The **input** comes as array of two strings. The first element is the target string and the second element is the text in which to search for occurrences.

The **output** should be a number, printed on the console.

#### Examples

Input	Output
['the', 'The quick brown fox jumps over the lay dog.']	1
['ma', 'Marine mammal training is the training and caring for marine life such as, dolphins, killer whales, sea lions, walruses, and other marine mammals. It is also a duty of the trainer to do mental and physical exercises to keep the animal healthy and happy.']	7

### 4. Extract Text

You will be given a text as a string. Write a JS function that extracts and prints only the text that's surrounded by parentheses.

The **input** comes as array of a single string element.

The **output** is printed on the console on a single line, in the form of a comma-separated list.

#### Examples

Input
['Rakiya (Bulgarian brandy) is self-made liquor (alcoholic drink)']
Output
Bulgarian brandy, alcoholic drink

### 5. Aggregate Table

You will be given a list of towns and incomes for each town, formatted in a table, separated by pipes (|). Write a JS function that extracts the names of all towns and produces a sum of the incomes. Note that splitting may result in empty string elements and the number of spaces may be different in every table.

The **input** comes as array of string elements. Each element is one row in a formatted table.

The **output** is printed on the console on two lines. On the first line, print a comma-separated list of all towns and on the second, the sum of all incomes.

#### Examples

Input
['  Sofia   300', '  Veliko Tarnovo   500', '  Yambol   275']
Output

## 6. Restaurant Bill

You are tasked to write a JS function that receives an array of purchases and their prices and prints all your purchases and their total sum.

The **input** comes as an array of string elements – the elements on even indexes (0, 2, 4...) are the product names, while the elements on odd indexes (1, 3, 5...) are the corresponding prices.

The **output** should be printed on the console - a single sentence containing all products and their total sum in the format “**You purchased {all products separated by comma + space} for a total sum of {total sum of products}**”.

### Examples

Input
['Beer Zagorka', '2.65', 'Tripe soup', '7.80', 'Lasagna', '5.69']
Output
You purchased Beer Zagorka, Tripe soup, Lasagna for a total sum of 16.14

Input
['Cola', '1.35', 'Pancakes', '2.88']
Output
You purchased Cola, Pancakes for a total sum of 4.23

## 7. Usernames

Write a JS function that parses a list of emails and returns a list of usernames, generated from them. Each username is composed from the name of the email address, a period and the first letter of every element in the domain name. See the examples for more information.

The **input** comes as array of string elements. Each element is an email address.

The **output** is printed on the console on a single line as a comma-formatted list.

### Examples

Input
['peshoo@gmail.com', 'todor_43@mail.dir.bg', 'foo@bar.com']
Output
peshoo.gc, todor_43.mdb, foo.bc

## 8. Censorship

The thought police are at it again and they need your help! Write a JS function that would **censor news articles**. You will be given a text and then a list of strings that need to be blacked out from the text. Replace all occurrences of the strings with dashes of the same

length as the string. The strings will **not overlap**, so order of processing is not important. See the examples for more information.

The **input** comes as array of string elements. The first element is the text to be censored and all following elements are the strings to be censored.

The **output** is the return value of your functions. Save the censored results in a string and return it.

## Examples

Input
['roses are red, violets are blue', ', violets are', 'red']
Output
roses are ----- blue

Input
['David Ruben Piqtoukun (born 1950) is an Inuit artist from Paulatuk, Northwest Territories. His output includes sculpture and prints; the sculptural work is innovative in its use of mixed media. His materials and imagery bring together modern and traditional Inuit stylistic elements in a personal vision. An example of this is his work "The Passage of Time" (1999), which portrays a shaman in the form of a salmon moving through a hole in a hand. While shamanic imagery is common in much of Inuit art, the hand in this work is sheet metal, not a traditional material such as walrus ivory, caribou antler or soapstone. Ruben\'s brother, Abraham Apakark Anghik Ruben, is also a sculptor. Fellow Inuit artist Floyd Kuptana learned sculpting techniques as an apprentice to David Ruben.', 'Inuit']
Output
David Ruben Piqtoukun (born 1950) is an ----- artist from Paulatuk, Northwest Territories. His output includes sculpture and prints; the sculptural work is innovative in its use of mixed media. His materials and imagery bring together modern and traditional ----- stylistic elements in a personal vision. An example of this is his work "The Passage of Time" (1999), which portrays a shaman in the form of a salmon moving through a hole in a hand. While shamanic imagery is common in much of ----- art, the hand in this work is sheet metal, not a traditional material such as walrus ivory, caribou antler or soapstone. Ruben's brother, Abraham Apakark Anghik Ruben, is also a sculptor. Fellow ----- artist Floyd Kuptana learned sculpting techniques as an apprentice to David Ruben.

## 9. Escaping

You will be given a list of strings, containing user-submitted data. Write a JS function that prints an HTML list from the data. The strings, however, may contain special HTML characters, which is an oft-used method for injection attacks. To prevent unwanted behavior or harmful content, all special characters need to be replaced with their encoded counterparts – they will look the same to the user, but will not pose a security risk. Use the following table to compose your function:

Raw	Encoded
<	&lt;
>	&gt;
&	&amp;

"	&quot;
---	--------

Use the provided HTML template to visually test your code – if you don’t escape the control characters, formatted HTML will show up. Don’t care how the HTML template works. Your job is to write the JS escaping function only.

The **input** comes as array of string elements.

The **output** is the return value of your function. Compose the list in a string and return it. See the examples for formatting details.

HTML
<pre> &lt;!DOCTYPE html&gt; &lt;html lang="en"&gt; &lt;head&gt;   &lt;meta charset="UTF-8"&gt;   &lt;title&gt;Escaping&lt;/title&gt; &lt;/head&gt; &lt;body&gt; &lt;div&gt;&lt;label for="userInput"&gt;Paste test input here:&lt;/label&gt;&lt;/div&gt; &lt;div&gt;   &lt;textarea rows="12" cols="40" id="userInput"&gt;&lt;/textarea&gt;   &lt;input type="button" value="Escape"     onclick="document.getElementById('result').innerHTML = htmlEscape(JSON.parse(document.getElementById('userInput').value.replace(/'/g, String.fromCharCode(34))));"/&gt; &lt;/div&gt; &lt;div&gt;&lt;label for="result"&gt;Results will show up here:&lt;/label&gt;&lt;/div&gt; &lt;div id="result"&gt;&lt;/div&gt; &lt;script&gt;   function htmlEscape(input) {     // TODO   } &lt;/script&gt; &lt;/body&gt; &lt;/html&gt; </pre>

## Examples

Input
['<b>unescaped text</b>', 'normal text']
Output
<pre> &lt;ul&gt;   &lt;li&gt;&amp;lt;b&gt;unescaped text&lt;/b&gt;&lt;/li&gt;   &lt;li&gt;normal text&lt;/li&gt; &lt;/ul&gt; </pre>

Input
['<div style=\"color: red;\">Hello, Red!</div>', '<table><tr><td>Cell 1</td><td>Cell 2</td><tr>']
Output
<pre> &lt;ul&gt;   &lt;li&gt;&amp;lt;div style=\"color: red;\"&gt;Hello, Red!&lt;/div&gt;&lt;/li&gt; </pre>

```
<li>&lt;table&gt;&lt;tr&gt;&lt;td&gt;Cell 1&lt;/td&gt;&lt;td&gt;Cell  
2&lt;/td&gt;&lt;tr&gt;</li>  
</ul>
```

## 10. Match All Words

Write a JS function that matches all words in a text, a word is anything that consists of letters, numbers or underscores (\_).

The **input** comes as an array of string consisting of a single entry - the text from which to extract the words.

The **output** should be printed on the console and should consist of all words concatenated with a "|" (pipe), check the examples bellow to better understand the format.

### Examples

Input
['A Regular Expression needs to have the global flag in order to match all occurrences in the text']
Output
A Regular Expression needs to have the global flag in order to match all occurrences in the text

Input
['_(Underscores) are also word characters']
Output
_ Underscores are also word characters

### Hints

- Read about the special characters in Regular Expressions at MDN to find some that can ease your task [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular\\_Expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions)

## 11. Simple Email Validation

Write a JS function that validates simple emails. The emails should have a **username**, which consists only of **English alphabet letters** and **digits**, a **"@" sign**, and a domain name after it. The domain should consist **only of 2 strings separated by a single dot**. The 2 strings should contain **NOTHING** but **lowercase English alphabet letters**.

The **input** comes as an array of strings. The array will hold one element which is an email.

The **output** should be printed on the console. If the given email is valid, print **"Valid"**, if it is not, print **"Invalid"**.

### Examples

Input	Output
valid@email.bg	Valid

## 12. \*Expression Split

Write a JS function that splits a passed in JS code into separate parts. The passed in code will always have one or more spaces between operators and operands. Normal brackets ('(', ')'), commas (,), semicolons (;) and the member access operator ('.'(**dot**), as in "console.log") should also be used for splitting. String literals will always be initialized with double quotes (") and will **contain only letters**. Make sure there are no empty entries in the output.

The **input** comes as array of one string element - the JS code that has to be split.

The **output** should be printed on the console, with each elements obtained from the split is printed on a new line.

### Examples

Input	Output
['let sum = 4 * 4,b = "wow";']	let sum = 4 * 4 let b = "wow"
['let sum = 1 + 2;if(sum > 2){\tconsole.log(sum);}']	let sum = 1 + 2 if sum > 2 { console log sum }

## 13. Match the Dates

Write a JS function that finds and extracts all the dates in the given sentences. The dates should be in format

**d-MMM-yyyy. Example: 12-Jun-1999, 3-Dec-2017.**

The **input** comes as an array of strings. Each string represents a sentence.

The **output** should be printed on the console. The output should consist of all extracted **VALID** dates. Each element should be printed on a new line.

## Examples

Input
I am born on 30-Dec-1994. This is not date: 512-Jan-1996. My father is born on the 29-Jul-1955.
Output
30-Dec-1994 (Day: 30, Month: Dec, Year: 1994) 29-Jul-1955 (Day: 29, Month: Jul, Year: 1955)

Input
1-Jan-1999 is a valid date. So is 01-July-2000. I am an awful liar, by the way - Ivo, 28-Sep-2016.
Output
1-Jan-1999 (Day: 1, Month: Jan, Year: 1999) 28-Sep-2016 (Day: 28, Month: Sep, Year: 2016)

## 14. Parse the Employee Data

Write a JS function that **validates employee data**, and stores it **if it is valid**. The employee data consists of 3 elements - **employee name**, **employee salary** and **employee position**.

The **input** comes as an array of strings. Each element represents input employee data. You should capture only the valid from them. The input will have the following format:

**{employeeName} - {employeeSalary} - {employeePosition}**

The **Employee name** will be a **string**, which can contain only **English alphabet letters** and must **start with a capital**. The **Employee salary** should be a **VALID number**. The **employee position** can contain **English alphabet letters, digits, dashes, and can consist of several words**. Any input that **does NOT follow** the specified above rules, is to be treated as **invalid**, and is to **be ignored**.

The **output** should be printed on the console. For every **valid employee data** found, you should print each of its elements. Check the examples.

## Examples

Input
Isacc - 1000 - CEO Ivan - 500 - Employee Peter - 500 - Employee
Output
Name: Isacc Position: CEO Salary: 1000  Name: Ivan Position: Employee Salary: 500



Name: Peter  
Position: Employee  
Salary: 500

Input
Jonathan - 2000 - Manager Peter- 1000- Chuck George - 1000 - Team Leader
Output
Name: Jonathan Position: Manager Salary: 2000  Name: George Position: Team Leader Salary: 1000

## Hints

- Use **Groups** for this problem, it would be a lot easier.

## 15. Form Filler

Write a JS function that automatically fills a form for a lazy client. The client will give you **3 elements** of **data** about himself – his **username**, his **email**, and his **phone number**. After those 3 elements you will be given the form, as text, with several placeholders in it. You must replace each **valid placeholder** with its corresponding value. The placeholders have special symbols and can **contain only English alphabet letters**. There are **3 types** of valid placeholders:

- `<!{letters}!>` - put the given username in place of this
- `<@{letters}@>` - put the given email in place of this
- `<+{letters}+>` - put the given email in place of this

The **input** comes as an array of strings. The **first 3 elements** will represent – the **username**, the **email** and the **phone number**. Each element after that will represent a sentence, if you find a placeholder somewhere in those sentences you should replace it.

The **output** should be printed on the console. The output should consist of all sentences, printed again, this time with their placeholders replaced with the actual data.

## Examples

Input
Pesho pesho@softuni.bg 90-60-90 Hello, <!username!>! Welcome to your Personal profile. Here you can modify your profile freely. Your current username is: <!fdsfs!>. Would you like to change that? (Y/N) Your current email is: <@DasEmail@>. Would you like to change that? (Y/N) Your current phone number is: <+number+>. Would you like to change that? (Y/N)

Output
Hello, Pesho! Welcome to your Personal profile. Here you can modify your profile freely. Your current username is: Pesho. Would you like to change that? (Y/N) Your current email is: pesho@softuni.bg. Would you like to change that? (Y/N) Your current phone number is: 90-60-90. Would you like to change that? (Y/N)

## 16. \*Match Multiplication

You are given a text with **numbers** multiplied by **\*** in format **{num1} \* {num2}**. Your job is to extract each two numbers in the above format, multiply them and replace them with their product. The **first number** is integer, can be negative. The **second number** is integer or floating-point and can be negative. There could be whitespace around the **"\*"** symbol.

The **input** comes as an array of strings, holding only one element – the text holding the numbers.

The **output** should be printed on the console – it consists of the same text with the multiplied numbers replaced by their product.

### Examples

Input
My bill: <b>2*2.50</b> (beer); <b>2* 1.20</b> (kepab); <b>-2 * 0.5</b> (deposit).
Output
My bill: <b>5</b> (beer); <b>2.4</b> (kepab); <b>-1</b> (deposit).
Input

### Hint

- Match the numbers to be multiplied by regex with groups. Check the overloads for the **String.replace** function, there may be an overload with a **callback** that can help you.