Exercise: 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/1480/.

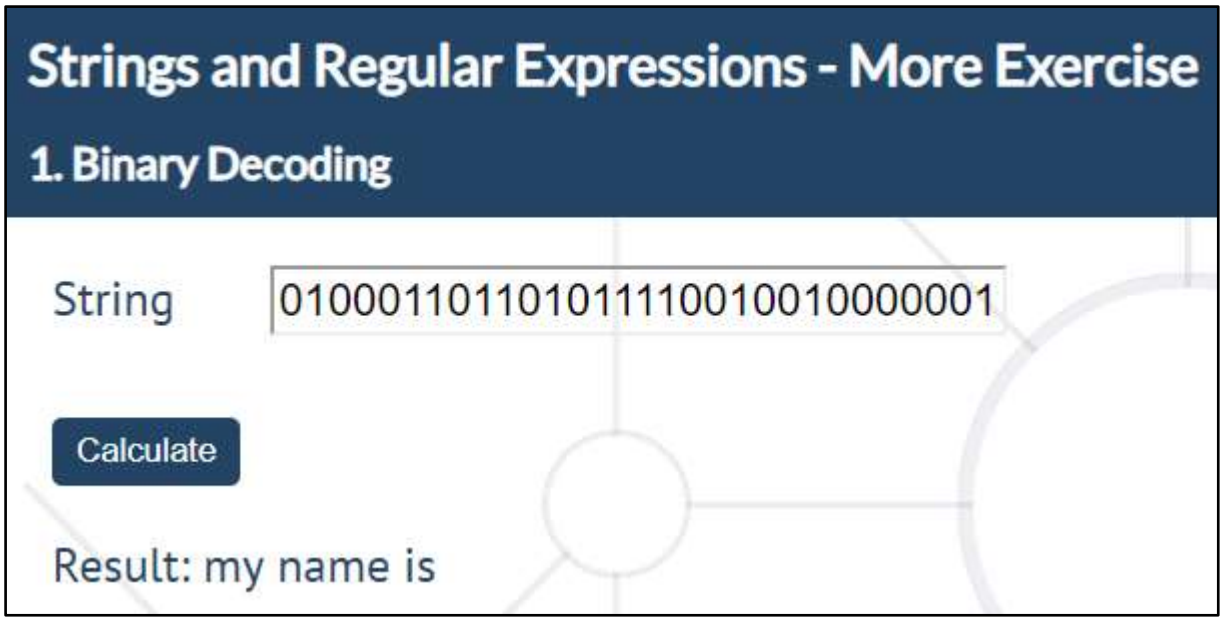
1. Binary Decoding

You will receive a string from zeros and ones. Write a function that converts it into text. To do that, you need to calculate the weight of the string (sum of all the ones). The result of that sum should be a single digit (e.g '10011101111100111111001101111' here the sum is 21, but since we need only one digit, we sum 2 and 1 and get 3). After that, you need to remove from the start and the end the calculated sum and split the remaining string into groups of 8 characters. Then convert each segment of binary code into decimals to get the ascii code of each element. Then print the result in the "result" span (only the letters and spaces).

Example

Input	Output	Comment
'0100011011010111100100100000011011100110000101101101011001010010000001101001011100110010'	my name is	The sum here is 40; 4 + 0 = 4; we take 4 characters from the beginning and end, then we split it into segments of 8 characters and we get '01101101', '01111001', '00100000', '01101110', '01100001', '01101101', '01100101', '00100000', '01101001', '01110011'. When we convert each one of these segments into ascii we get 109 121 32 110 97 109 101 32 150 115. When we convert that into text we get 'my name is'
'0100111001101101111011001100111010001110101011011100110100100100000011100110111010001110100101100111001010110111001110100010'	softuni student	

Hints



2. Expedition problems

The expedition is over and everyone has returned successfully to the rest house. It turns out, however, that one from the group has fallen behind. He has sent a message to the leader but his device is broken and his message contains unwanted symbols, which prevent it from being read. Since the leader does not understand anything from programming, he has assigned the task of decrypting to you.

You will receive a text (**string**), which can contain **all of the ASCII symbols**, **including new lines** and **tabs**. The location of the lost person and his message must be retrieved from this string. The text contains a **keyword** that indicates the **beginning** and the **end** of the **message**. The geographical coordinates come as **a pair** of longitude ("**east**") and latitude ("**north**") and each coordinate should meet the following conditions:

1. It should start with "**north**"/"**east**", **case-insensitive**;
2. Next come **2 digits**, which form the **whole part of the degrees**;
3. The whole part of the degrees is separated from the decimal part by "**,**" and there may be any number of characters between them, **except** ","
4. The **decimal part** consists of **6 digits**

In case there is **more than one** longitude or latitude in the text, take **the latter**. The message is surrounded by the **keyword**, which will be the **first argument** from the input. The second argument from the input will be the **text**, containing **both the location** and **the message** of the lost person. See the examples below to understand how it works.

Input

The first argument contains the keyword and the second argument contains the text. There will always be **at least one pair of coordinates**.

Output

Print the latitude in a **paragraph** in the **"result"** span in the following format:

<degrees>.<decimal part> **N**

Print the second in a **paragraph** in the **"result"** span in the following format:

<degrees>.<decimal part> **E**

On the last line print the message:

Message: *<message>*

Example

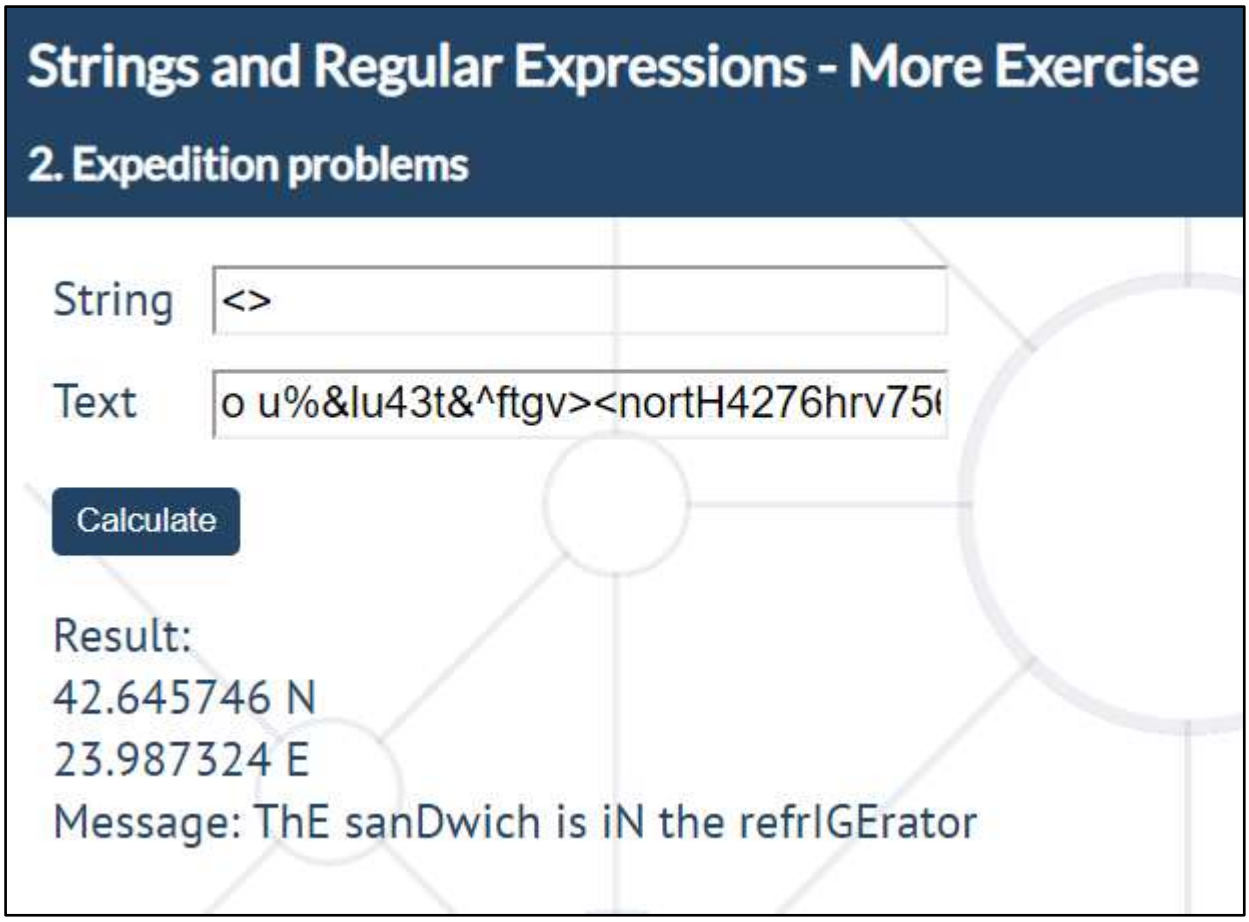
Note there are three instances of north coordinates – the first two are ignored and only the last one is

Input
<> o u%&lu43t&^ftgv><northH4276hrv756dcc, jytbu64574655k <>ThE sanDwich is iN the refrIGErator<>yl i75evEAsTer23,lfwe 987324tlblu6b
Output
42.645746 N 23.987324 E Message: ThE sanDwich is iN the refrIGErator

counted.

Input
4ds eaSt 19,432567noRt north east 53,123456north 43,3213454dsNot all those who wander are lost.4dsnorth 47,874532
Output
47.874532 N 19.432567 E Message: Not all those who wander are lost.

Hints



3. James Bond

You are the spy master of a guild of spies, since you're all carefully watched, your spies communicate with you by leaving **encoded messages**. You will receive a **special key** and **lines of text** which you must comb in order to find the **encoded messages**.

The special key will consist only of **one or more English letters**.

- The special key must be preceded by either a **space** or the **start of the string** in order to be considered valid in the string.
- The special key may appear in **any casing** (its letters could be a random mix of lower and uppercase) in the **lines of text**.

Valid encoded messages must meet these requirements:

- An **encoded message** must immediately follow the **special key**, being separated from it only by **one or more spaces**.
- An encoded message must be **at least 8 symbols long** and consist only of the symbols **!**, **%**, **$**, **#** or **Capital English letters**.

- The encoded message must be followed by a **space**, a **dot (.)**, a **comma (,)** or the **end of the string**.

After finding the correct **encoded messages**, you must **decode** and **replace them in the original text**. The decoding should be as follows:

- The symbol **!** becomes the number **1**
- The symbol **%** becomes the number **2**
- The symbol **#** becomes the number **3**
- The symbol **$** should become the number **4**
- **Capital English letters** should become their **lower case counterparts**

Constraints

- A pair of a **special key** and **encoded message** will never be split between multiple lines.
- The preceding space before a **special key** will never overlap with a trailing space after an **encoded message**.
- There will never be an **encoded message** equal to the **special key**.

Input

The **input** comes as an array of strings - the first element is the **special key**, each element after it is a **line of text**.

Output

The **output** should be displayed in the DOM - consisting of the entire text with the **correct encoded messages** replaced with their **decoded** versions, each **line of text** in a new paragraph in the "result" span.

Examples

Input
specialKey In this text the specialKey HELLOWORLD! is correct, but the following specialKey $HelloWorl#d and spEcIaLKEy HOLLOWORLD1 are not, while SpeCIaLkeY SOM%%ETH$IN and SPECIALKEY ##$$##$$ are!

Output
In this text the specialKey helloworld1 is correct, but the following specialKey $HelloWorl#d and spEcIaLKEy HOLLOWORLD1 are not, while SpeCIaLkeY som22eth4in and SPECIALKEY 33443344 are!

Input
enCode Some messages are just not encoded what can you do? RE - ENCODE THEMNOW! - he said. Damn encode, ITSALLHETHINKSABOUT, eNcoDe BULL$#!%.
Output
Some messages are just not encoded what can you do? RE - ENCODE themnow1 - he said. Damn encode, ITSALLHETHINKSABOUT, eNcoDe bull4312.

Hints

4. Airport Check

Your next task is to extract information about a flight.

You will be given a string in two parts separated by a comma. The **first** one is a **string** you need to extract the information from. The **second** one will be the **information you need to print**.

The information to extract is as follows:

Passenger name:

- Starts with a **space** (' ')
- Consists of **2 or 3 names** (**upper and lower-case letters**)
- The names should be **separated by a single dash ('-')**
- Each name should **start with an upper-case letter**
- If there are **3 names**, **the second one should end with a dot ('.')**
- If there are **2 names**, **the second should not contain any dots**
- Ends with a **space (' ')**
- **Valid** examples with **2 names**: **' Test-Testov ', ' V-N '**
- **Valid** examples with **3 names**: **`' T-T.-Testov ', ' Valid-V.-Name '`**

Airport:

- Starts with a **space** (' ')
- Should consist of **3 upper-case letters**, the symbol '**/**' and **3 more upper-case letters**
- Ends with a **space (' ')**
- **Valid** airports are: **`' SOF/VAR ', ' VIE/AIR '`**

Flight number:

- Starts with a **space (' ')**
- Followed by **1 to 3 upper-case letters**
- Followed by **1 to 5 digits**
- Ends with a **space (' ')**
- **Valid** flight numbers are: **`' DWF24 ', ' S43159 ', ' OE314 '`**

Company:

- Starts with dash and space **`'- '`**
- Followed by **one or more letters** (**starting with an upper-case letter**)
- Followed by **`'*'`**
- Followed by **one or more letters** again (**starting with an upper-case letter**)
- Ends with a space **(' ')**
- **Valid** companies: **`'- Wizz*Air ', '- X*Y '`**

Input

String in two parts separated by a comma: the string you have to extract the information from and the command for the output

Output

4 different types of outputs depending on the **second** parameter:

`'name'` – print **`'Mr/Ms, {name}, have a nice flight!'`**

`'flight'` – print **`'Your flight number {flightNumber} is from {fromAirport} to {toAirport}.'`**

`'company'` – print **`'Have a nice flight with {companyName}.'`**

`'all'` – print **`'Mr/Ms, {name}, your flight number {flightNumber} is from {fromAirport} to {toAirport}. Have a nice flight with {company}.'`**

The name should be printed **without the `'-'`** (if any) and **with spaces instead**:
`' STEF-T.-Stefanov ' -> 'STEF T. Stefanov'`.

The **company** should be printed **without the `'- '`** and the **`'*'`**:

`' - Wizz*Air ' -> 'Wizz Air'.`

The **flight number** should be printed **without the spaces in the front and back**:

`' OS806 ' -> 'OS806'`

`Display the output in the "result" span`

Examples

Input	Output
`ahah Second-Testov )*))&&ba SOF/VAR ela** FB973 - Bulgaria*Air -opFB900 pa-SOF/VAr//_- T12G12 STD08:45  STA09:35 , all`	`Mr/Ms, Second Testov, your flight number FB973 is from SOF to VAR. Have a nice flight with Bulgaria Air.`
`TEST-T.-TESTOV  SOF/VIE OS806 - Austrian*Airlines T24G55 STD11:15 STA11:50 , flight`	`Your flight number OS806 is from SOF to VIE.`