Exercises: Data Types, Expressions, Statements

Problems for exercises and homework for the "JavaScript Fundamentals" course @ SoftUni.
Submit your solutions in the SoftUni Judge system at https://judge.softuni.bg/Contests/308.

1. Hello, JavaScript!

Write a JS function that can receive a name as input and print a greeting to the console.

The **input** comes as array of strings with one element that is the name of the person.

The **output** should be printed to the console.

Examples

Input	Output
[Pesho]	Hello, Pesho, I am JavaScript!
[Bill Gates]	Hello, Bill Gates, I am JavaScript!

Hints

Since input comes from an array of values, and we only want one value, we can capture it easily in a variable, directly at the function's declaration.

```
function solve([name]) {...}
```

The code above is identical to capturing the entire input array and then extracting just the index we need and storing it in a named variable:

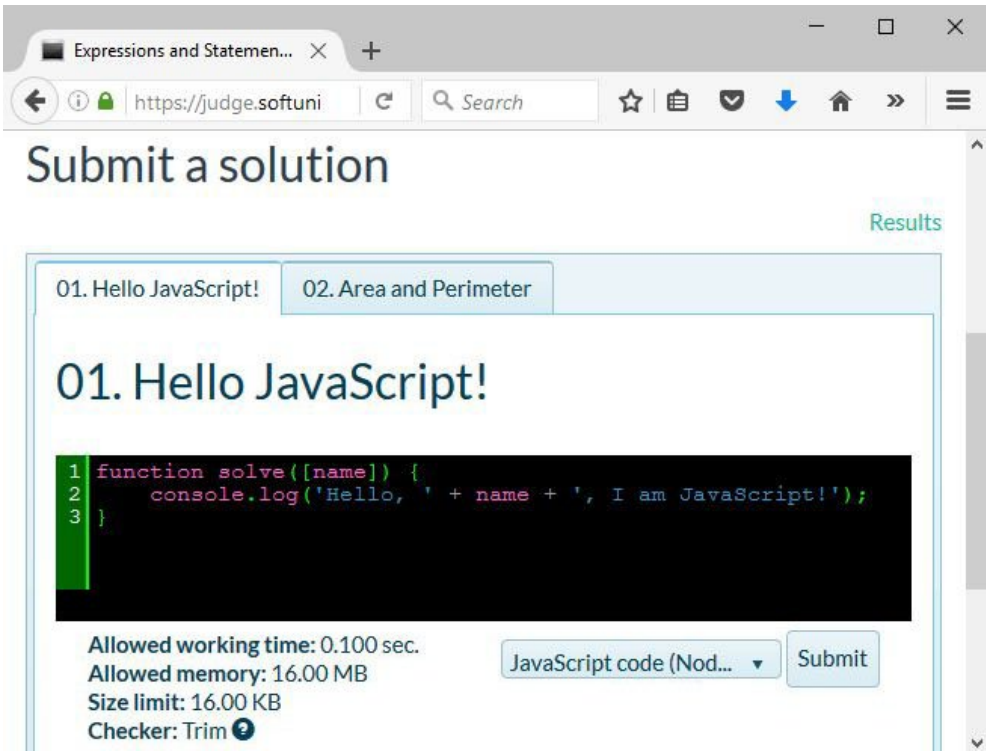
```
function solve(input) {
    let name = input[0];

    {...}
}
```

We need to concatenate three strings – the two static parts of our greeting and the name of the person in the middle. We can do this by simply adding the three strings with the addition operator. Since this is an operation which returns the concatenated string, we can directly perform this expression in a call to **console.log()**. Note the space at the end of the first string:

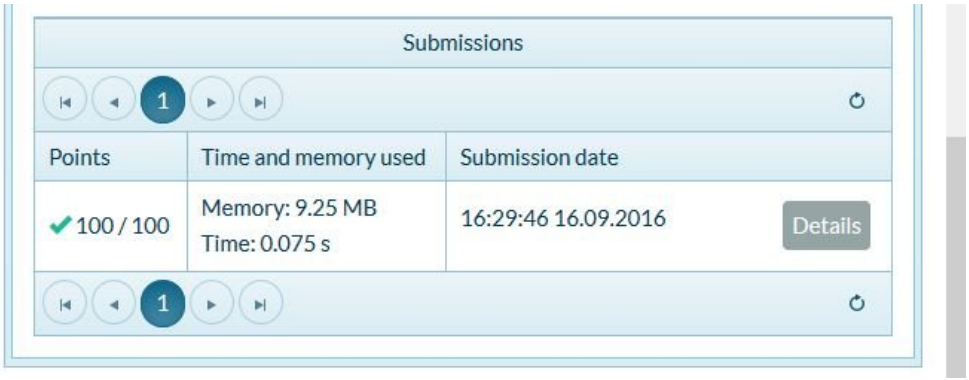
```
console.log('Hello, ' + name + ', I am JavaScript!');
```

You should be ready to submit your solution to the **judge system**. Open the judge contest for this homework and submit your code: https://judge.softuni.bg/Contests/308. It should look like

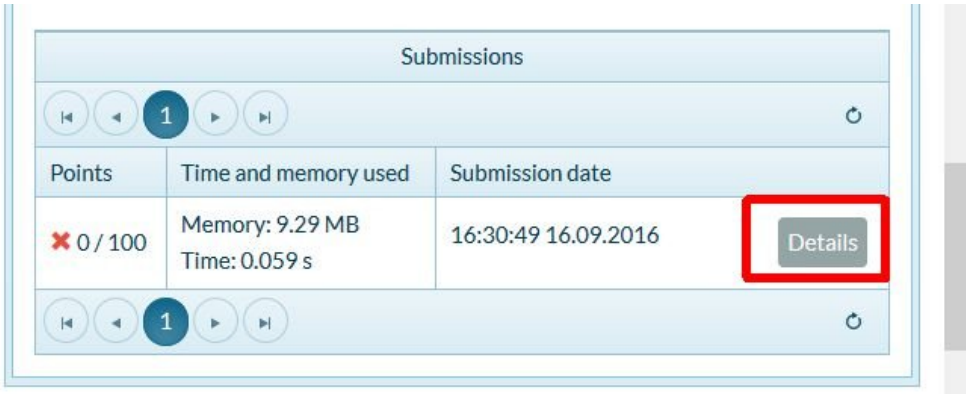
this:



The judge system should **accept your solution** as correct:



If you get an error, you can see what went wrong in the detailed report, using the highlighted button:



Here you can see what the system expected and what your program's result was. You can also see what the input for the test was:

Note you can only view detailed information for zero tests – they do not give you points, but are handy for finding typos and debugging some errors. Chances are, if you manage to get all zero tests to pass, you'll also have some competitive tests passing too.

2. Area and Perimeter

Write a JS function that calculates the area and perimeter of a rectangle by given two sides.

The **input** comes as array of strings that need to be parsed as numbers.

The **output** should be printed to the console on two lines.

Examples

Input	Output
[2, 2]	4 8
[1, 3]	3 8
[2.5, 3.14]	7.85 11.28

Hints

The multiplication operator will automatically coerce the input variables to numbers, so we can directly find the area of the rectangle by multiplying the two input elements. Addition, however, works on strings by joining them end to end, as we saw in the previous task. In order to avoid such behavior, we need to explicitly convert the input strings to number. This can be accomplished using the **Number(numberAsString)** function, built into JavaScript:

```
function solve([a, b]) {
    a = Number(a);
    b = Number(b);

    {...}
}
```

The remaining operations are straightforward arithmetic and finally printing the two results (area and perimeter) to the console.

3. Distance over Time

Two objects start from point ***A*** and travel in the same direction at constant speeds V_1 and V_2 for a period ***T***. Write a JS function that calculates the distance between the two object at the end of the period.

The **input** comes as array of strings that need to be parsed as numbers. The first two elements are the speeds to the two objects in km/h and the third element is the time in seconds.

The **output** should be printed to the console. Calculate the distance in meters.

Examples

Input	Output
[0, 60, 3600]	60000
[11, 10, 120]	33.33333333333337
[5, -5, 40]	111.11111111111111

Hints

Speed, time and distance are related to each other with the following formula:

$$S = V * T$$

However, the incoming units need to be equalized first and after the calculation, a final conversion needs to be done to match the required output units. There are 3600 seconds in an hour and 1000 meters in a kilometer. We don't know which object covered a greater distance, so simply subtracting them from one another may result in a **negative number**. Distance however is absolute (always positive), so we need to get the absolute value of the operation, using the built-in **Math.abs(number)** functions:

```
let delta = Math.abs(dist1 - dist2);
```

4. Distance in 3D

Write a JS function that calculates the distance between the two points in 3D by given coordinates.

The **input** comes as an array of strings that need to be parsed as numbers. The first three elements are the x, y and z coordinates for the first point and the second set of arguments are the coordinates of the other point.

The **output** should be printed to the console.

Examples

Input	Output
[1, 1, 0, 5, 4, 0]	5
[3.5, 0, 1, 0, 2, -1]	4.5

Hints

You can find the horizontal and vertical offset between two points by calculating the difference between their respective coordinates. Use Pythagoras' theorem to find the distance.

5. Grads to Degrees

Land surveyors use grads (also known as gon, 400 grads in a full turn) in their documents. Grads are rather unwieldy though, so you need to write a JS function that converts from grads to degrees. In addition, your program needs to constrain the results within the range $0° \leq x < 360°$, so if you arrive at a value like -15°, it needs to be converted to 345° and 420° becomes 60°.

The **input** comes as an array of one string element that needs to be parsed as a number.

The **output** should be printed to the console.

Examples

Input	Output
[100]	90

Input	Output
[400]	0

Input	Output
[850]	45

Input	Output
[-50]	315

Hints

You can use the remainder (modulo) operator to get a value that is cyclic – it returns the same result for all input values with offset equal to the second parameter. For instance, **n % 10** will return 3 with values for **n** 3, 13, 23, 243, 1003 and so on.

6. Compound Interest

Write a JS function that calculates the total accumulated value for a monetary deposit by given principal sum, interest rate, compounding frequency and overall length.

The **input** comes as an array of strings that need to be parsed as a numbers. The first value is the principal sum, the second is the interest rate in percent, the third is the compounding period in months and the final value is the total timespan, given in years.

The **output** should be printed to the console, with two decimal places.

Examples

Input	Output
[1500, 4.3, 3, 6]	1938.84

Input	Output
[100000, 5, 12, 25]	338635.49

Hints

The formula for calculating compound interest is as follows:

$$F = P\left(1 + \frac{i}{n}\right)^{nt}$$

where:

- ***P*** is the principal sum
- ***i*** is the nominal interest rate

- ***n*** is the compounding frequency
- ***t*** is the overall length of time the interest is applied

Note that at the beginning of the task you are given the compounding period, which is inversely related to the frequency. You need to express the frequency as how many times in a year the interest is compounded. For instance, a 3-month period means the interest will be updated 4 times in a year. Any percentages need to be expressed as a fraction.

7. *Rounding

Write a JS function that rounds numbers to specific precision.

The **input** comes as an array of two string elements that need to be parsed as a numbers. The first value is the number to be rounded and the second is the precision (significant decimal places). If a precision is passed, that is more than **15** it should automatically be reduced to **15**.

The **output** should be printed to the console. Do not print insignificant decimals.

Examples

Input	Output
[3.1415926535897932 384626433832795, 2]	3.14

Input	Output
[10.5, 3]	10.5

8. Imperial Units

Imperial units are confusing, but still in use in some backwards countries (Myanmar, Liberia and the United States are the only countries still using them). They are so confusing in fact, that native users struggle to convert between them. Write a JS function that converts from inches to feet and inches. There are 12 inches in a foot. See the example for formatting details.

The **input** comes as an array of one string element that needs to be parsed as a number.

The **output** should be printed to the console.

Examples

Input	Output
[36]	3'-0"

Input	Output
[55]	4'-7"

Input	Output
[11]	0'-11"

9. Now Playing

Write a JS function that displays information about the currently playing musical track.

The **input** comes as an array of string elements. The first element is the name of the track, the second is the name of the artist performing and the third is the duration in minutes and seconds.

The **output** should be printed to the console in the following format:

Now Playing: {artist name} - {track name} [{duration}]

Examples

Input	Output

['Number One', 'Nelly', '4:09']	Now Playing: Nelly – Number One [4:09]

10. Compose Tag

Write a JS function that composes an HTML image tag.

The **input** comes as an array of string elements. The first element is the location of the file and the second is the alternate text.

The **output** should be printed to the console in the following format:

<img src="{file location}" alt="{alternate text}">

Examples

Input	Output
['smiley.gif', 'Smiley Face']	<img src="smiley.gif" alt="Smiley Face">

11. Binary to Decimal

Write a JS function that reads an 8-bit binary number and converts it to a decimal.

The **input** comes as an array of one string element, representing a binary number.

The **output** should be printed to the console.

Examples

Input	Output
['00001001']	9

Input	Output
['11110000']	240

12. Assign Properties

Write a JS function that composes an object by given properties. There will be 3 sets of property-value pairs (a total of 6 elements). Assign each value to its respective property of an object and return the object as a result of the function.

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

The **output** should be returned as a value.

Examples

Input
['name', 'Pesho', 'age', '23', 'gender', 'male']
Output

{
  name: 'Pesho',
  age: '23',
  gender: 'male'
}

Input
['ssid', '90127461', 'status', 'admin', 'expires', '600']
Output

{
ssid: '90127461',
status: 'admin',
expires: '600'
}

13. *Last Month

Write a JS function that receives a date as array of strings containing **day, month** and **year** in that order. Your task is to print the last day of previous month (the month **BEFORE** the given date). Check the examples to better understand the problem.

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

The **output** should be a single number representing the **last day** of the previous month.

Examples

Input	Output
['17','3','2002']	28

Input	Output
['13','12','2004']	30