EuroSkills 2025 National Competition - HUNGARY

Speed Test

Task 1

Write a program to determine if a string contains all unique characters. Return true if it does and false otherwise.

The string may contain any of the 128 ASCII characters.

Specification

```
hasUniqueChars(str)
```

Parameters

`str`: `string` - The string that may or may not contain all unique characters

Return Value

`boolean` - True if all characters in the string are unique

Examples

str	Return Value
"abcdefg"	true
"abbcdefg"	false

Task 2

Consider an array where each element in the array contains a positive integer digit. Taken as a whole, such an array represents a positive integer number. The rightmost position of the array represents the least significant digit of the number.

An example digit array is `[4, 2]` which represents the integer `42`.

In this challenge, you will write a function to increment the number in the digit array by 1. For example, `upArray([4, 2])` will return the array `[4, 3]`.

Here is the complete specification for `upArray`:

`upArray(arr)`

increases the digit array value by one

Parameters

`arr`: `Array<number>` - an array of integers to be increased.

Return Value

`Array<number>` - an array with the new value.

Constraints

- Parameter array will not be empty
- Array will only contain non-negative single-digit integers
- Array will not contain leading zeroes unless its length is exactly 1

Examples

arr	Return Value
[5,7,4]	[5,7,5]

[4,3,9]	[4,4,0]
[9]	[1,0]

Task 3

Write a function that receives two strings and returns the number of characters we would need to rotate the first string forward to match the second.

For instance, take the strings "fatigue" and "tiguefa". In this case, the first string can be rotated 5 characters forward to produce the second string, so 5 would be returned. Here are the steps:

```
no rotations: "fatigue"
1st rotation: "efatigu"
2nd rotation: "uefatig"
3rd rotation: "guefati"
4th rotation: "iguefat"
5th rotation: "tiguefa"
```

If the second string isn't a valid rotation of the first string, the method should return -1.

Specification

`shiftedDiff(first, second)`

computes the number of rotations to make string `first` equal to string `second`, if possible

Parameters

`first`: **`string`** - string to be rotated

`second`: **`string`** - target string to be matched by rotating `first`

Return Value

`number` - Number of rotations needed to turn string `first` into `second`, `-1` if invalid

Examples:

```
"coffee", "eecoff" => 2
"eecoff", "coffee" => 4
"moose", "Moose" => -1
"isn't", "'tisn" => 2
"Esham", "Esham" => 0
"dog", "god" => -1
```

Task 4

Background
Deoxyribonucleic acid (DNA) is a chemical found in the nucleus of cells and carries the "instructions" for the development and functioning of living organisms.

DNA is created by two strands of *nucleotides* that are bonded together in complementary pairs. For each *base* on one side, there is an opposite *base* on the other side. There are 4 symbols used to represent the *bases*, `A`, `T`, `C`, and `G`.

Symbols `A` and `T` are complements of each other, as are `C` and `G`.

Task
You have a function with one side of the DNA, and you need to get the other complementary side. The DNA strand may be empty if there is no DNA at all. In this case, you can simply return the empty string.

Specification

```
dnaComplement(dna)
```

Parameters
`dna`: **`string`** - DNA strand

Return Value
`string` - A new string generated by returning the complement of the input strand.

Constraints
It will always be a string, but it might be empty.

It will never be `null/nil` or `undefined`.

Examples

dna	Return Value
`"A"`	`"T"`
`"T"`	`"A"`
`"C"`	`"G"`
`"G"`	`"C"`

`"ATTGC"`	`"TAACG"`
`""`	`""`

Task 5

Usually when you buy something, you're asked whether your credit card number, phone number or answer to your most secret question is still correct. However, since someone could look over your shoulder, you don't want that shown on your screen. Instead, we mask it.

Your task is to write a function `maskify`, which changes all but the last four characters into `'#'`.

Examples

Input	Output	Comments
`"4556364607935616"`	`"############5616"`	
`"64607935616"`	`"#######5616"`	
`"1"`	`"1"`	No `#`s if less than 4 characters
`""`	`""`	Make sure to handle empty strings
`"Skippy"`	`"##ippy"`	

Documentation

maskify(cc)

Parameters:

cc: `String`

A string of any characters.

Guaranteed Constraints:

- The input string will never be `null` or `undefined`.

Returns: `String`

The input string with all but the last four characters replaced with `'#'`.

Task 6

A Cartesian coordinate system is a coordinate system that specifies each point uniquely in a plane by a pair of numerical coordinates, which are the signed distances to the point from two fixed perpendicular directed lines, measured in the same unit of length.

The coordinates of a point in the grid are written as (x, y). Each point in a coordinate system has eight neighboring points. It is provided that the grid step = 1.

Your task is to write a function that takes a coordinate on the x-axis and y-axis and returns a list of all the neighboring points. Points inside your result list don't have to be sorted--any ordering is valid.

Specification

```
cartesianNeighbor(x, y)
```

find all surrounding coordinates of given point

Parameters

`x`: **`number`** - Coordinate for point on x-axis

`y`: **`number`** - Coordinate for point on y-axis

Return Value

`Array<Array<number>>` - An array of coordinates surrounding provided point

Examples

x	y	Return Value
2	2	`[[1,1],[1,2],[1,3],[2,1],[2,3],[3,1],[3,2],[3,3]]`
-17	7	`[[-18,6],[-18,7],[-18,8],[-17,6],[-17,8],[-16,6],[-16,7],[-16,8]]`

Task 7

Given a non-negative integer, return an array / a list of the individual digits in order.

Specification

```
digitize(n)
```

separate multiple digit numbers into an array

Parameters

`n`: **`number`** - Number to be converted

Return Value

`Array<number>` - Array of separated single digit integers

Examples

n	Return Value
`123`	`[1,2,3]`
`8675309`	`[8,6,7,5,3,0,9]`

Task 8

You have two friends, **Mortis** and **Darryl** are trying to say something to you at the same time. Their words have been combined to a sentence and each word is separated by a space.

Words that are all uppercase represent Mortis's word and words that are all lowercase represent Darryl's word. Words that include upper and lower cases is some random sound. You have to try to understand what Mortis and Darryl are trying to say to you.

Output the sentence that Mortis is trying to say with the first character uppercase and the rest lowercase. Do the same thing with Darryl's sentence. If one of them didn't say anything output "*None*" string instead.

Examples

Input	Output
`"AA. Bb cc."`	`{M: "Aa", D: "Cc."}`
`"AA. Bb"`	`{M: "Aa", D: "None"}`
`"Bb cc."`	`{M: "None", D: "Cc."}`

Documentation

separateSentences(sentence)

Parameters:

sentence: `String`

A string of characters.

Guaranteed Constraints:

- The string will never be `null` or `undefined`.
- The string may be empty (`""`), and may contain any alphabetic character, or symbols.

Returns: `Object`

Returns `Object` with property's M, and D for the mentioned characters. Each value contains their sentence.

Task 9

Complete the function in your editor.

Your function must merge strings `a` and `b`, and then return a single merged string. A merge operation on two strings is described as follows:

- Append alternating characters from `a` and `b`, respectively, to some new string, *mergedString*.
- Once all of the characters in one of the strings have been merged, append the remaining characters in the other string to *mergedString*.

Specification

```
mergeStrings(a, b)
```

Parameters

a: **string**

b: **string**

Return Value

string - The *mergedString*

Constraints

`1 ≤ |a length|, |b length| ≤ 25000`

Examples

a	b	Return Value
`"abc"`	`"def"`	`"adbecf"`
`"ab"`	`"def"`	`"adbef"`
`"abc"`	`"de"`	`"adbec"`

Task 10

Background

Markdown is a formatting syntax used by many documents (these instructions, for example!) because of its plain-text simplicity and its ability to be translated directly into HTML.

Task

Let's write a simple markdown parser function that will take in a single line of markdown and be translated into the appropriate HTML. To keep it simple, we'll support only one feature of markdown in atx syntax: headers.

Headers are designated by ***(1-6)*** hashes followed by a space, followed by text, meaning `<h7>` is not a valid header. The number of hashes determines the header level of the HTML output.

Specifications

`markdownParser(markdown)`

Transforms given string into correct header form

Parameters

`markdown`: **`string`** - String to be changed into markdown format

Return Value

`string` - Formatted string

Examples

markdown	Return Value
`"# Header"`	`"<h1>Header</h1>"`
`"## Header"`	`"<h2>Header</h2>"`
`"###### Header"`	`"<h6>Header</h6>"`

Additional Rules

Header content should only come after the initial hashtag(s) plus a space character.

Invalid headers should just be returned as the markdown that was received, no translation is necessary.

Spaces before and after both the header content and the hashtag(s) should be ignored in the resulting output.

Task 11

You are going to be given a word. Your job is to return the middle character of the word. If the word's length is odd, return the middle character. If the word's length is even, return the middle 2 characters.

```
getMiddle(s)
```

Find the middle character(s) of a word.

Parameters

`s`: **`string`** - word to pull middle characters from

Return Value

`string` - letter(s) in the middle of the word

Constraints

`0 < str < 1000`

Examples

s	Return Value
`"test"`	`"es"`
`"testing"`	`"t"`
`"middle"`	`"dd"`

Task 12

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Complete the function `solution` so that it returns the sum of all the multiples of 3 or 5 **below** the number passed in.

Specification

```
solution(number)
```

Finds the sum of multiples of 3 or 5 that is less than the provided number

Parameters

`number`: **`number`** - Maximum number to check against

Return Value

`number` - Sum of all multiples of either 3 or 5

Examples

number	Return Value
10	23
200	9168

Task 13

Remove the vowels from a string. Your task is to write a function that takes a string and returns a new string with all vowels removed.

For example, the string "Almost finished with this challenge!" would become "lmst fnshd wth ths chllng!".

Specification

```
disemvowel(str)
```

Removes all vowels from a given string

Parameters

`str`: **`string`** - A string to be stripped of its vowels

Return Value

`string` - A string that has been disemvoweled

Examples

str	Return Value
`"Be the change you want to see in the world - Gandhi"`	`"B th chng y wnt t s n th wrld - Gndh"`
`"A person who never made a mistake never tried anything new - Einstein"`	`" prsn wh nvr md mstk nvr trd nythng nw - nstn"`

Task 14

Write a function that takes a string of parentheses and determines if the parentheses are valid. This function should return true if the string is valid, and false if it's invalid.

Parentheses are valid if every opening parenthesis has a paired closing parentheses that occurs at a higher index in the string.

Specification

`validParentheses(braces)`

Checks if the parentheses order is valid

Parameters

`braces`: **`string`** - A string representation of an order of parentheses

Return Value

`boolean` - Returns `true` if order of parentheses are valid

Examples:

Input	Output
`"()"`	true
`")(()))"`	false
`"("`	false
`"(())((()())())"`	true

All input strings will be nonempty, and will only consist of open parentheses `"("` and/or closed parentheses `")"`.

Task 15

In a factory a printer prints labels for boxes. For one kind of boxes the printer has to use colors which, for the sake of simplicity, are named with letters from `a` to `m`.

The colors used by the printer are recorded in a control string. For example a "good" control string would be

`aaabbbbhaijjjm` meaning that the printer used three times color `a`, four times color `b`, then one time color `a` … etc.

Sometimes there are problems: lack of colors, technical malfunction and a "bad" control string is produced e.g. `aaaxbbbbyyhwawiwjjjwwm`. Note the `x` and `w`s.

You have to write a function which given a string will output the error rate of the printer as a **string** representing a rational whose numerator is the number of errors and the denominator the length of the control string.

Don't reduce this fraction to a simpler expression.

The input string has a length greater or equal to one and contains only letters from `a`to `z`.

Examples:

Input	Output
`"aaabbbbhaijjjm"`	`"0/14"`
`"aaaxbbbbyyhwawiwjjjwwm"`	`"8/22"`

Documentation

printerErrors(s)

Parameters:

s: `String`

A string of letters representing the colors the printer needs to use.

Guaranteed Constraints:

- The input will always be a `string`.
- It will always contain at *least* one letter.
- It will never be `null` or `undefined`.
- The string will only contain the letters from `a` to `z`.

Returns: `String`

A string containing the number of errors (letters `n` to `z`) and a number of total control characters, separated by a `/`. For `3` errors out of `5` characters, the result would be `3/5`.

The resulting fraction will not be reduced to a simpler expression.