# Exercises: Recursive Functions

Problems for exercises and homework for the "Functional Programming“ course from the official "Applied Programmer“ curriculum.

You can check your solutions here: <https://judge.softuni.org/Contests/3539/Recursive-Functions-Exercises>.

## Sum of Natural Even Numbers

Your task is to **calculate** the **sum** of **natural** **even** **numbers** from 1 to n using the **recursion** **technique.**Write your solution in summing() function.

### Input

You will receive **one** **number** – n (the **last** **number** we need to **sum** **up**).

### Output

Your function should **return** the final result as string - the **sum** of natural **even** **numbers**.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| summing(10); | 30 |
| summing(15); | 56 |

In Judge submit only summing() function.

## Decimal to Binary

Your task is to **turn** the **number** into its **binary** equivalent. Write your solution in a **recursive function** decimalToBinary().

### Input

You will receive **one** **number** – the **number** you **should** **transform**.

### Output

Your function should **return** the final result as string – the **binary** **equivalent** of number.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| decimalToBinary(10); | 1010 |
| decimalToBinary(15); | 1111 |

## Logarithm Second of N

Define a **function** logSecond() that accepts **one** **parameter** - a **number** and **returns** as a **result** a **log2(n)** of the **given** **number** (**rounded** to **integer** **type**).

### Input

You will receive **one parameter** -the **number** forour **logarithm**.

### Output

The logSecond() function should return **logarithm** **second** of the **given** **number** (**rounded** to **integer** **type**) as string.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| logSecond(10); | 3 |
| logSecond(15); | 3 |
| logSecond(10000); | 13 |

### Hints

1. Define the **function** logSecond() as for **1** let it **return** the **result** **0**.
2. For n > 1 return **result** **1** + **result** of **recursive** **call** of the **same** **function** for n / 2.

* In this way a **cyclic** **effect** is achieved (similar to for-loop), as the **beginning** is n - the **condition** is n > 1, and at each iteration n **decreases** **twice**.

## Least Common Multiple

Write a JS **function** lcm() which find the **Least Common Multiple** (LCM) of **two** **positive** **numbers**. Learn more about LCM in Wikipedia: <https://en.wikipedia.org/wiki/Least_common_multiple>.

### Input

You will receive **array** of **two** **positive** **numbers**.

### Output

Your function should **return** the **final** **result** as string – **Least Common Multiple**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| lcm([20, 4]); | 20 |
| lcm([200, 2]); | 200 |

In **Judge** submit only lcm() function.

## Factorial (Tail Recursion)

Create a function factorial() that takes the **number n** as a parameter and **returns n factorials** as the result **(5! = 5\*4\*3\*2\*1)**.Use a **tail recursion algorithm**!

### Input

You will receive onenumber – n - **factorial** **number**.

### Output

The function should **return** the final result as string - **factorial**.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| factorial(5); | 120 |
| factorial(10); | 3628800 |

In **Judge** submit only the factorial() function. It might have another tail-recursive function inside.

## Fibonacci Sequence (Tail Recursion)

Create a fibonacci() **function** that takes the number **n** as a parameter and returns the **nth** number in the **Fibonacci sequence**.  
The Fibonacci sequence **starts** with **1** and each subsequent number is equal to the **sum** of the **previous two.**

* The **second** number in the sequence is equal to 1 + 0 **(for a zero number in the series is 0)**
* The **third** number in the series is equal to the second + first (1 + 1) = 2, etc.

Use a **tail recursion algorithm**!

**Think about negative cases!**

### Input

You will receive one number – n – the beginning of Fibonacci sequence.

### Output

Your function should **return** the final result as string – the result of Fibonacci sequence.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| fibonacci(10); | 55 |
| fibonacci(-21); | 10946 |

In **Judge** submit only fibonacci() function.

## Inverted Triangle

Create a **recursive** **function** invertedTriangle() that accept **one** **parameter** - n and **prints** an **inverted** **triangle** of \* on the **console**, **starting** with n **stars** on the **first** **line** and **printing** **one** **less** on **each** **subsequent** **line**.

At **input 0**, print **a single space** (" ") on the console.

### Input

You will receive one number – n – the **number** of **stars** for **first** **row**.

### Output

The invertedTriangle() function should **return** the final result as string – the **inverted** **triangle**.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| invertedTriangle(5); | \*\*\*\*\*  \*\*\*\*  \*\*\*  \*\*  \* |
| invertedTriangle(1); | \* |
| invertedTriangle(4); | \*\*\*\*  \*\*\*  \*\*  \* |

In **Judge** submit only invertedTriangle() function.

## Palindromes

Your task is to write a **recursive function** isPalindrome() that **checks** if a **string** is a **palindrome**.  
**Palindrome** - a **word**, **phrase**, or **sequence** that reads the same **backward** as **forward**. For example, "madam", "noon", "eve", and "level" are **palindromes**.

### Input

You will receive **one word – string**, which you need to check.

### Output

* If the **word** is **palindrome** your function should return true.
* Else if the **word** is **not** **palindrome** your function should return false.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| isPalindrome('eve'); | true |
| isPalindrome('no0n'); | false |

In Judge submit only isPalindrome() function.