# Functions and Statements - Exercise

Problems with exercise and homework for the ["JS Front-End" Course @ SoftUni.](https://softuni.bg/modules/132)

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/3789/Functions-and-Statements-Exercises>

## Smallest of Three Numbers

Write a function that receives **three integers** and prints the **smallest** number. Use an appropriate name for the function.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2,  5,  3 | 2 |
| 600,  342,  123 | 123 |
| 25,  21,  4 | 4 |
| 2,  2,  2 | 2 |

## Add and Subtract

You will receive **three** **integer numbers.**

Write a function sum() to calculate the sum of the first **two** integers and a function **subtract()**, which subtracts the result of the function the sum() and the **third** integer.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 23,  6,  10 | 19 |
| 1,  17,  30 | -12 |
| 42,  58,  100 | 0 |

## Characters in Range

Write a function that receives **two characters** and prints on a single line all the characters in between them according to the **ASCII** code. Keep in mind that the second character code might be **before** the first one inside the **ASCII table**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'a',  'd' | b c |
| '#',  ':' | $ % & ' ( ) \* + , - . / 0 1 2 3 4 5 6 7 8 9 |
| 'C',  '#' | $ % & ' ( ) \* + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B |

## Odd and Even Sum

You will receive a **single number.** You have to write a function, that returns the **sum** of **all even** and **all odd** digits from that number.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1000435 | Odd sum = 9, Even sum = 4 |
| 3495892137259234 | Odd sum = 54, Even sum = 22 |

## Palindrome Integers

A palindrome is a number, which reads the same **backward as forward**, such as 323 or 1001. Write a function, which receives an **array of positive integers** and checks if each integer is a palindrome or not.

### Output

* If the current integer is a palindrome, print: **"true"**
* Otherwise, print: **"false"**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [123,323,421,121] | false  true  false  true |  | [32,2,232,1010] | false  true  true  false |

### Hints

* Read more about palindromes: <https://en.wikipedia.org/wiki/Palindrome>

## Password Validator

Write a function that checks if a given password is valid. Password validations are:

* The **length** should be **6 - 10** characters (inclusive)
* It should consist **only of** **letters** and **digits**
* It should have **at least 2** digits

If a password is a valid print: **"Password** **is** **valid"**.

If it is **NOT** valid, for every unfulfilled rule print a message:

* **"Password must be between 6 and 10 characters"**
* **"Password must consist only of letters and digits"**
* **"Password must have at least 2 digits"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'logIn' | Password must be between 6 and 10 characters  Password must have at least 2 digits |
| 'MyPass123' | Password is valid |
| 'Pa$s$s' | Password must consist only of letters and digits  Password must have at least 2 digits |

## NxN Matrix

Write a function that receives a single integer number **n** and prints **nxn** matrix with that number.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 | 3 3 3  3 3 3  3 3 3 |
| 7 | 7 7 7 7 7 7 7  7 7 7 7 7 7 7  7 7 7 7 7 7 7  7 7 7 7 7 7 7  7 7 7 7 7 7 7  7 7 7 7 7 7 7  7 7 7 7 7 7 7 |
| 2 | 2 2  2 2 |

## Perfect Number

Write a function that receives a **number** and checks if that number is perfect or NOT.

A perfect number is a **positive** integer that is equal to the **sum** ofits **proper positive divisors**. That is the sum of its positive **divisors** excluding the number itself (also known as its **aliquot sum**).

### Output

* If the number is perfect, print: **"We have a perfect number!"**
* Otherwise, print: **"It's not so perfect."**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 6 | We have a perfect number! | 1 + 2 + 3 |
| 28 | We have a perfect number! | 1 + 2 + 4 + 7 + 14 |
| 1236498 | It's not so perfect. |  |

### Hint

Equivalently, a perfect number is a number that is **half the sum** of all of its positive divisors (including itself) => 6 is a perfect number because it is the sum of 1 + 2 + 3 (all of which are divided without residue).

* Read about the Perfect number here: <https://en.wikipedia.org/wiki/Perfect_number>

## Loading Bar

You will receive a **single number** between **0** and **100**, which is divided with 10 without residue (0, 10, 20, 30...).

Your task is to create a function that visualizes a **loading bar** depending on the number you have received in the input.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 30 | 30% [%%%.......]  Still loading... |
| 50 | 50% [%%%%%.....]  Still loading... |
| 100 | 100% Complete!  [%%%%%%%%%%] |

## Factorial Division

Write a function that receives **two** integer numbers. Calculate the **factorial** of each number. Divide the first result by the second and print the division formatted to the **second decimal** point.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5,  2 | 60.00 |  | 6,  2 | 360.00 |

### Hints

* Read more about factorial here: <https://en.wikipedia.org/wiki/Factorial>
* You can use [recursion](https://en.wikipedia.org/wiki/Recursion_(computer_science))