

[Learning](#)[Chat](#)[Gitea](#)

Name - Login

Quest06

[Subject](#)[1 Solution](#)[Additional
Resources \(3\)](#)

Quest06

Remember to git add && git commit && git push each exercise!

We will execute your function with our test(s), please DO NOT PROVIDE ANY TEST(S) in your file

For each exercise, you will have to create a folder and in this folder, you will have additional files that contain your work. Folder names are provided at the beginning of each exercise under `submit directory` and specific file names for each exercise are also provided at the beginning of each exercise under `submit file(s)`.

Quest06	My Iterative Pow
Submit directory	ex00
Submit file	my_iterative_pow.c

Description

Control Center

[Group formation](#)[In Progress](#)[Submitted](#)[Test review](#)**Finished: approved**[Go To DoCode](#)

Access:

READ

WRITE

[Go To Gitea](#)[Keep Working On This Solution](#)**Also working on
the project**

$2^2 \Rightarrow 4$
 $2^3 \Rightarrow 8$
 $2^4 \Rightarrow 16$

Let's create a function to calculate the pow of a number!

Create an iterated function that returns the value of a power applied to a number. An power lower than 0 returns `0`.

Overflows don't have to be handled.

First parameter is the number, second parameter is the `power`.
You have to use a `loop` (for/while/...) to perform this exercise

Function prototype (c)

```
/*  
**  
** QWASAR.IO -- my_iterative_pow  
**  
** @param {int} param_1  
** @param {int} param_2  
**  
** @return {int}  
**  
*/  
  
int my_iterative_pow(int param_1, int  
param_2)  
{  
  
}
```

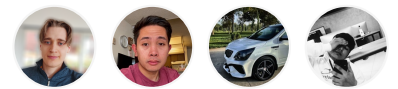
Example 00

Input: 2 && 2
Output:
Return Value: 4

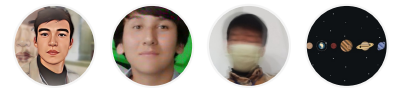
Example 01

Input: 2 && 3
Output:
Return Value: 8

the project



[lucas_v](#) [medina](#) [alimbo](#) [abdura](#)
[i](#) [t](#) [ev_a](#) [hm_o](#)



[barato](#) [toirov](#) [erkinov](#) [vusaro](#)
[v_b](#) [s](#) [_d](#) [v_d](#)

Just finished



[ahmur](#) [xolxoja](#) [ergash](#) [atabek](#)
[ato_x](#) [y_f](#) [ev_a](#) [ov_a](#)



[sobirjo](#)
[n_h](#)

Type

Project

Group
Size

1
Participant

Review
system

Test Review (Gandalf)

Difficult
y

Initiation

Average
duration

1
Week

Project's Metadata

Example 02

Input: 2 && 4
Output:
Return Value: 16

Quest06	My Recursive Pow
Submit directory	ex01
Submit file	my_recursive_pow.c

Description

$2^2 \Rightarrow 4$
 $2^3 \Rightarrow 8$
 $2^4 \Rightarrow 16$

Let's create a function to calculate the pow of a number!

Create an iterated function that returns the value of a power applied to a number. An power lower than 0 returns 0.

Overflows don't have to be handled.

First parameter is the number, second parameter is the power
You have to use the recursive method to perform this exercise.
While / for / any loop are forbidden.

Function prototype (c)

Project's metadata

Project

id: 37

name: quest06

visible: True

```
/*
**
** QWASAR.IO -- my_recursive_pow
**
** @param {int} param_1
** @param {int} param_2
**
** @return {int}
**
*/

int my_recursive_pow(int param_1, int
param_2)
{

}
```

Example 00

Input: 2 && 2
Output:
Return Value: 4

Example 01

Input: 2 && 3
Output:
Return Value: 8

Example 02

Input: 2 && 4
Output:
Return Value: 16

Tip

Google the following: recursive programming

Quest06	My Iterative Factorial
Submit directory	ex02
Submit file	my_iterative_factorial.c

Description

$2! \Rightarrow 2 \times 1 \Rightarrow 2$

$3! \Rightarrow 3 \times 2 \times 1 \Rightarrow 6$

$4! \Rightarrow 4 \times 3 \times 2 \times 1 \Rightarrow 24$

Let's create a function to calculate the **factorial** of a number!

Create an iterated function that returns a number. This number is the result of a factorial operation based on the number given as a parameter.

If there's an error, the function should return **0**.

You have to use a **loop** (for/while/...) to perform this exercise

Function prototype (c)

```
/*
**
** QWASAR.IO -- my_iterative_factorial
**
** @param {int} param_1
**
** @return {int}
**
*/

int my_iterative_factorial(int param_1)
{

}
```

Example 00

Input: 2

Output:

Return Value: 2

Example 01

Input: 3
Output:
Return Value: 6

Example 02

Input: 4
Output:
Return Value: 24

Quest06	My Recursive Factorial
Submit directory	ex03
Submit file	my_recursive_factorial.c

Description

$2! \Rightarrow 2 \times 1 \Rightarrow 2$

$3! \Rightarrow 3 \times 2 \times 1 \Rightarrow 6$

$4! \Rightarrow 4 \times 3 \times 2 \times 1 \Rightarrow 24$

Let's create a function to calculate the **factorial** of a number!

Create an iterated function that returns a number. This number is the result of a factorial operation based on the number given as a parameter.

If there's an error, the function should return **0**.

You have to use the recursive method to perform this exercise. While / for / any loop are forbidden.

Function prototype (c)

```
/*  
**  
** QWASAR.IO -- my_recursive_factorial  
**  
** @param {int} param_1  
**  
** @return {int}  
**  
*/  
  
int my_recursive_factorial(int param_1)  
{  
  
}
```

Example 00

Input: 2
Output:
Return Value: 2

Example 01

Input: 3
Output:
Return Value: 6

Example 02

Input: 4
Output:
Return Value: 24

Tip

Google the following: recursive programming

--	--

Quest06	My Atoi
Submit directory	ex04
Submit file	my_atoi.c

Description

The atoi() function in C takes a string (which represents an integer) as an argument and returns its value of type int. So basically the function is used to convert a string argument to an integer.

Syntax:

```
int atoi(const char strn)
```

Parameters: The function accepts one parameter strn which refers to the string argument that is needed to be converted into its integer equivalent.

Return Value: If strn is a valid input, then the function returns the equivalent integer number for the passed string number. If no valid conversion takes place, then the function returns zero.

Function prototype (c)

```
/*
**
** QWASAR.IO -- my_atoi
**
** @param {char*} param_1
**
** @return {int}
**
*/

int my_atoi(char* param_1)
{

}
```

Example 00

Input: "2"
Output:
Return Value: 2

Example 01

Input: "123"
Output:
Return Value: 123

Example 02

Input: "-10"
Output:
Return Value: -10

Tip

(In C)

Split the number by dividing it, and to get the rest are you aware of the mod operator?

You should google it :)

Quest06	My Fibonacci
Submit directory	ex05
Submit file	my_fibonacci.c

Description

Create a function `my_fibonacci` that returns the `n-th` element of the Fibonacci sequence, the first element being at the 0 index. We'll consider that the Fibonacci sequence starts like this: 0, 1, 1, 2.

If the `value` is less than 0, the function should return -1.

It should be prototyped:

Recursive will be helpful here.

Google fibonacci.

Function prototype (c)

```
/*  
**  
** QWASAR.IO -- my_fibonacci  
**  
** @param {int} param_1  
**  
** @return {int}  
**  
*/  
  
int my_fibonacci(int param_1)  
{  
  
}
```

Example 00

Input: 2
Output:
Return Value: 1

Example 01

Input: 3
Output:
Return Value: 2

Example 02

Input: 4
Output:
Return Value: 3

Tip

Google the following: recursive programming

