### Chapter III

Exercise 00: ft\_iterative\_factorial

	Exercice: 00	
/	ft_iterative_factorial	
Turn-in directory : $ex00/$		
Files to turn in : ft_iterative	e_factorial.c	/
Allowed functions: Nothing		/
Remarks : n/a		

- 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.
- Here's how it should be prototyped :

int ft\_iterative\_factorial(int nb);

### Chapter IV

# Exercise 01: ft\_recursive\_factorial

Exercice: 01	
ft_recursive_factorial	/
Turn-in directory : $ex01/$	
Files to turn in : ft_recursive_factorial.c	
Allowed functions: Nothing	
Remarks: n/a	

- Create a recursive function that returns the factorial of the number given as a parameter.
- If there's an error, the function should return 0.
- Here's how it should be prototyped :

int ft\_recursive\_factorial(int nb);

### Chapter V

## Exercise 02: ft\_iterative\_power

Exercice: 02	
ft_iterative_power	
Turn-in directory: $ex02/$	
Files to turn in : ft_iterative_power.c	/
Allowed functions: Nothing	
Remarks: n/a	

- 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.
- Here's how it should be prototyped :

```
int ft_iterative_power(int nb, int power);
```

## Chapter VI

Exercise 03: ft\_recursive\_power

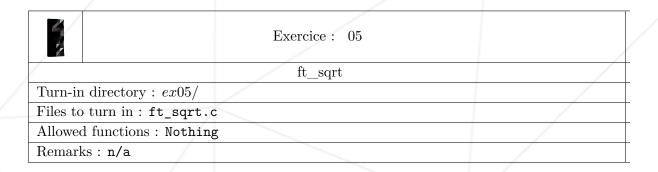
	Exercice: 03	
	ft_recursive_power	
Turn-in directory : $ex03/$		
Files to turn in : ft_recur	sive_power.c	
Allowed functions: Nothin	ng	
Remarks: n/a		

- Create a recursive function that returns the value of a power applied to a number.
- Same conditions as before.
- Here's how it should be prototyped :

int ft\_recursive\_power(int nb, int power);

# Chapter VIII

Exercise 05: ft\_sqrt

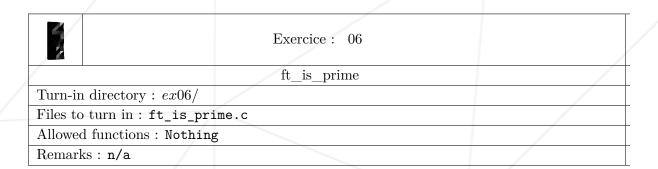


- Create a function that returns the square root of a number (if it exists), or 0 if the square root is an irrational number.
- Here's how it should be prototyped :

int ft\_sqrt(int nb);

## Chapter IX

Exercise 06: ft\_is\_prime



- Create a function that returns 1 if the number given as a parameter is a prime number, and 0 if it isn't.
- Here's how it should be prototyped :

int ft\_is\_prime(int nb);

• Your function must return its result in less than two seconds.



0 and 1 are not prime numbers.

### Chapter X

Exercise 07: ft\_find\_next\_prime

Exercice: 07	
ft_find_next_prime	
Turn-in directory : $ex07/$	
Files to turn in : ft_find_next_prime.c	
Allowed functions: Nothing	
Remarks: n/a	

- Create a function that returns the next prime number greater or equal to the number given as argument.
- Here's how it should be prototyped:

int ft\_find\_next\_prime(int nb);