Chapter III

Exercise 00 : ft_iterative_factorial

	Exercice : 00
ft_iterative_factorial	
Turn-in directory : *ex*00/	
Files to turn in : `ft_iterative_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);
```

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

Chapter IV

Exercise 01 : ft_recursive_factorial

	Exercice : 01
ft_recursive_factorial	
Turn-in directory : *ex*01/	
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 : *ex*02/	
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 :

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

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

Chapter VI

Exercise 03 : ft_recursive_power

	Exercice : 03
ft_recursive_power	
Turn-in directory : *ex*03/	
Files to turn in : `ft_recursive_power.c`	
Allowed functions : `Nothing`	
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

	Exercice : 05
ft_sqrt	
Turn-in directory : *ex*05/	
Files to turn in : `ft_sqrt.c`	
Allowed functions : `Nothing`	
Remarks : `n/a`	

- 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);
```

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

Chapter IX

Exercise 06 : ft_is_prime

	Exercice : 06
ft_is_prime	
Turn-in directory : *ex*06/	
Files to turn in : `ft_is_prime.c`	
Allowed functions : `Nothing`	
Remarks : `n/a`	

- 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 : *ex*07/	
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);
```

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