# Assignment 3

## Exercise 00: vc_iterative_factorial

| **Turn-in files** | **vc_iterative_factorial.c** |
|---|---|
| Allowed functions | Nothing |

- 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.
- Function prototype: `int iterative_factorial(int n);`

## Exercise 01: vc_recursive_factorial

| **Turn-in files** | **vc_recursive_factorial.c** |
|---|---|
| Allowed functions | Nothing |

- Create an recursive function that returns the factorial of th number given as a parameter.
- If there's an *error*, the function should return 0.
- Function prototype: `int vc_recursive_factorial(int n);`

## Exercise 02: vc_iterative_power

| **Turn-in files** | **vc_iterative_power.c** |
|---|---|
| Allowed functions | Nothing |

- 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.
- If there's an *error*, the function should return 0.
- Function prototype: `int iterative_power(int n, int power);`

## Exercise 03: vc_recursive_power

| **Turn-in files** | **vc_recursive_power.c** |
|---|---|
| Allowed functions | Nothing |

- Create a recursive function that returns the value of a power applied to a number.
- Same conditions as before.

- Function prototype: `int vc_recursive_power(int n, int power);`

## Exercise 04: vc_fibonacci

| Turn-in files | vc_fibonacci.c |
|---|---|
| Allowed functions | Nothing |

- Create a function ft_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.
- Your function should be recursive.
- If the n is less than 0, the function should return -1
- Function prototype: `int vc_fibonacci(int n);`

## Exercise 05: vc_sqrt

| Turn-in files | vc_sqrt.c |
|---|---|
| Allowed functions | Nothing |

- Create a function that returns the square root of a number (if it exists), or 0 if the square root is an irrational number.
- Your function must return its result in less than two seconds.
- Function prototype: `int vc_sqrt(int n);`

## Exercise 06: vc_is_prime

| Turn-in files | vc_is_prime.c |
|---|---|
| Allowed functions | Nothing |

- Create a function that returns 1 if the number given as a parameter is a prime number, and 0 if it isn't.
- Your function must return its result in less than two seconds.
- 0 and 1 are not prime numbers.
- Function prototype: `int vc_is_prime(int n);`

## Exercise 07: vc_find_next_prime

| Turn-in files | vc_find_next_prime.c |
|---|---|
| Allowed functions | Nothing |

- Create a function that returns the next prime number greater or equal to the number given as argument.

- Your function must return its result in less than two seconds.
- Function prototype: `int vc_find_next_prime(int n);`