B1 - Unix & C Lab Seminar

B-CPE-100

Day 05

Recursivity

1.0

EPITECH.

Day 05

language: C

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

- Don't push your `main` function into your delivery directory, we will be adding our own. Your files will be compiled adding our `main.c` and our `my_putchar.c` files.
- You are only allowed to use the `my_putchar` function to complete the following tasks, but **don't push it** into your delivery directory, and don't copy it in *any* of your delivered files.
- If one of your files prevents you from compiling with `*.c`, the Autograder will not be able to correct your work and you will receive a 0.

Create your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to `gitignore`.

All of the day's functions must produce an answer in under 2 seconds.
Overflows must be handled (as errors).

TASK 01 - MY_COMPUTE_FACTORIAL_IT

Delivery: my_compute_factorial_it.c
Write an iterative function that returns the factorial of the number given as a parameter.
It must be prototyped the following way:

```
int my_compute_factorial_it(int nb);
```

In case of error, the function should return 0.

`0! = 1`
if n < 0, `n! = 0`

TASK 02 - MY_COMPUTE_FACTORIAL_REC

Delivery: my_compute_factorial_rec.c
Write a recursive function that returns the factorial of the number given as a parameter.
It must be prototyped the following way:

```
int my_compute_factorial_rec(int nb);
```

In case of error, the function should return 0.

TASK 03 - MY_COMPUTE_POWER_IT

Delivery: my_compute_power_it.c
Write an iterative function that returns the first argument raised to the power `p`, where `p` is the second argument.
It must be prototyped the following way:

```
int my_compute_power_it(int nb, int p);
```

`n^0 = 1`
if p < 0, `n^p = 0`

TASK 04 - MY_COMPUTE_POWER_REC

Delivery: my_compute_power_rec.c
Write an recursive function that returns the first argument raised to the power p, where p is the second argument.
It must be prototyped the following way:

```
int my_compute_power_rec(int nb, int p);
```

TASK 05 - MY_COMPUTE_SQUARE_ROOT

Delivery: my_compute_square_root.c
Write a function that returns the square root (if it is a whole number) of the number given as argument.
If the square root is not a whole number, the function should return 0.
It must be prototyped the following way:

```
int my_compute_square_root(int nb);
```

TASK 06 - MY_IS_PRIME

Delivery: my_is_prime.c
Write a function that returns **1** if the number is prime and **0** if not.
It must be prototyped the following way:

```
int my_is_prime(int nb);
```

As you know, `0` and `1` are not prime numbers.

TASK 07 - MY_FIND_PRIME_SUP

Delivery: my_find_prime_sup.c
Write a function that returns the smallest prime number that is greater than, or equal to, the number given as a parameter.
It must be prototyped the following way:

```
int my_find_prime_sup(int nb);
```

TASK 08 - THE N QUEENS

Delivery: count_valid_queens_placements.c
Write a function that compute recursively and returns the number of possible ways to place `n` queens on a `n`x`n` chessboard without them being able to run into each other in a single move.
It must be prototyped the following way:

```
int count_valid_queens_placements(int n);
```

The output must be as follows:

```
Terminal
~/B-CPE-100> ./count_valid_queens_placements 1
1
~/B-CPE-100> ./count_valid_queens_placements 2
0
~/B-CPE-100> ./count_valid_queens_placements 3
0
~/B-CPE-100> ./count_valid_queens_placements 4
2
~/B-CPE-100> ./count_valid_queens_placements 5
10
```

Damn it, this is recursion day!