B1- C Pool

B-CPE-042

Day 05

Recursivity

v1.4

Day 05

Recursivity

repository name: : CPool_Day05
repository rights: : ramassage-tek
language: : C
group size: : 1

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- 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 0

Unit Tests

It is highly recommended to test your functions as you develop them. It is common practice to create a function named `main` (and a designated file to host it) to check the functions separately.
Create a directory named `tests`.
Create a `main` function within a file named `tests-$FUNCTION_NAME.c`, to be stored in the `tests` directory named.
This function must contain all the necessary calls to the task function in order to cover all of the function's possible situations (normal or irregular).

Task 1

my_fact_it

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

```
int my_fact_it(int nb);
```

In case of error, the function should return 0.
Delivery: CPool_Day05/my_fact_it.c

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

Task 2

my_fact_rec

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

```
int my_fact_rec(int nb)
```

In case of error, the function should return 0.
Delivery: CPool_Day05/my_fact_rec.c

Task 3

my_power_it

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

Delivery: CPool_Day05/my_power_it.c

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

Task 4

my_power_rec

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

Delivery: CPool_Day05/my_power_rec.c

Task 5

my_square_root

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_square_root(int nb);
```

Delivery: CPool_Day05/my_square_root.c

Task 6

my_is_prime

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);
```

Delivery: CPool_Day05/my_is_prime.c

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

Task 7

my_find_prime_sup

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);
```

Delivery: CPool_Day05/my_find_prime_sup.c

Task 8

The 8 queens v1

Write a function that returns the number of possibilities when placing 8 queens on a chessboard without them being able to run into each other in a single move.
The function must be prototyped as follows:

```
int my_8queens1();
```

Delivery: CPool_Day05/my_8queens/my_8queens1.c

A chessboard is composed of 8x8 squares.
A queen can move in rows, columns, and diagonally.
Hey, this is recursion day !

Task 9

The 8 queens v2

Write a function that displays every possible ways to place 8 queens on a chessboard without them being able to run into each other in a single move.
It must be prototyped the following way:

```
int my_8queens2();
```

The output will be as follows:

Terminal

```
~/B-CPE-042> ./8queens
15863724
16837425
17468253
36815724
```

Delivery: CPool_Day05/my_8queens/my_8queens2.c

Obviously, the above results are fake and only serve as an example of the display.
There is a line break after the last solution.
God damn it, this is recursion day !!