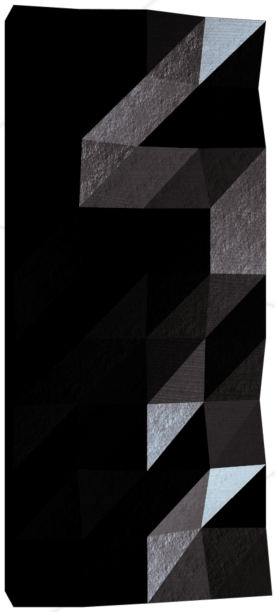

C Piscine

Day 04

Staff 42 pedago@42.fr

Summary: This document is the subject for Day04 of the C Piscine @ 42.

Contents

Chapter I

Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called `Norminator` to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass `Norminator`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- If ft_putchar() is an authorized function, we will compile your code with our `ft_putchar.c`.
- You'll only have to submit a main() function if we ask for a program.

- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses `gcc`.
- If your program doesn't compile, you'll get `0`.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on your right. Otherwise, try your peer on your left.
- Your reference guide is called `Google / man / the Internet / ....`
- Check out the "C Piscine" part of the forum on the intranet.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor ! Use your brain !!!

`Norminator must be launched with the -R CheckForbiddenSourceHeader flag. Moulinette will use it too.`

Chapter II

Foreword

Here are some lyrics extract from the `Harry Potter` saga:

```
Oh you may not think me pretty,
But don't judge on what you see,
I'll eat myself if you can find
A smarter hat than me.

You can keep your bowlers black,
Your top hats sleek and tall,
For I'm the Hogwarts Sorting Hat
And I can cap them all.

The Sorting Hat, stored in the Headmaster's Office.
There's nothing hidden in your head
The Sorting Hat can't see,
So try me on and I will tell you
Where you ought to be.

You might belong in Gryffindor,
Where dwell the brave at heart,
Their daring, nerve, and chivalry
Set Gryffindors apart;

You might belong in Hufflepuff,
Where they are just and loyal,
Those patient Hufflepuffs are true
And unafraid of toil;

Or yet in wise old Ravenclaw,
If you've a ready mind,
Where those of wit and learning,
Will always find their kind;

Or perhaps in Slytherin
You'll make your real friends,
Those cunning folks use any means
```

```
To achieve their ends.

So put me on! Don't be afraid!
And don't get in a flap!
You're in safe hands (though I have none)
For I'm a Thinking Cap!
```

Unfortunately, this subject's got nothing to do with the `Harry Potter` saga, which is too bad, because your exercises won't be done by `magic`.

Chapter III

Exercise 00 : ft_iterative_factorial

	Exercise 00
ft_iterative_factorial	
Turn-in directory : *ex*00/	
Files to turn in : `ft_iterative_factorial.c`	
Allowed functions : `None`	

- 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

	Exercise 01
ft_recursive_factorial	
Turn-in directory : $ex01/$	
Files to turn in : `ft_recursive_factorial.c`	
Allowed functions : `None`	

- 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

	Exercise 02
ft_iterative_power	
Turn-in directory : $ex02/$	
Files to turn in : `ft_iterative_power.c`	
Allowed functions : `None`	

- 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);
```

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

Chapter VI

Exercise 03 : ft_recursive_power

	Exercise 03
ft_recursive_power	
Turn-in directory : $ex03/$	
Files to turn in : `ft_recursive_power.c`	
Allowed functions : `None`	

- 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 VII

Exercise 04 : ft_fibonacci

	Exercise 04
ft_fibonacci	
Turn-in directory : *ex*04/	
Files to turn in : `ft_fibonacci.c`	
Allowed functions : `None`	

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

```
int ft_fibonacci(int index);
```

- Obviously, `ft_fibonacci` has to be recursive.
- If the `index` is less than 0, the function should return -1.

Chapter VIII

Exercise 05 : ft_sqrt

	Exercise 05
ft_sqrt	
Turn-in directory : *ex*05/	
Files to turn in : `ft_sqrt.c`	
Allowed functions : `None`	

- 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

	Exercise 06
ft_is_prime	
Turn-in directory : *ex*06/	
Files to turn in : `ft_is_prime.c`	
Allowed functions : `None`	

- 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

	Exercise 07
ft_find_next_prime	
Turn-in directory : $ex07/$	
Files to turn in : `ft_find_next_prime.c`	
Allowed functions : `None`	

- 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.

Chapter XI

Exercise 08 : The Eight Queens

	Exercise 08
The Eight Queens 1	
Turn-in directory : $ex08/$	
Files to turn in : `ft_eight_queens_puzzle.c`	
Allowed functions : `None`	

- The aim of this game is to place eight queens on a chessboard, without them being able to meet in one shot.
- Refresh your memories on chess rules.
- Evidently, recurstivity is required to solve this problem.
- Create a function that returns the number of possibilities to place those eight queens on the chessboard without them being able to reach each other.
- Here's how it should be prototyped :

```
int ft_eight_queens_puzzle(void);
```

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

Chapter XII

Exercise 09 : The Eight Queens 2

	Exercise 09
The Eight Queens 2	
Turn-in directory : $ex09/$	
Files to turn in : `ft_eight_queens_puzzle_2.c`	
Allowed functions : `ft_putchar`	

- Create a function that displays all possible placements of the eight queens on the chessboard, without them being able to reach each other.
- Recursivity is required to solve this problem.
- Here's how it should be prototyped :

```
void ft_eight_queens_puzzle_2(void);
```

- Here's how it'll be displayed :

```
$>./a.out
15863724
16837425
17468253
...
```

- The sequence goes from left to right. The first digit represents the first Queen's position in the first column (the index starting from 1). The Nth digit represents the Nth Queen's position in the Nth column.
- There's a line break after the last solution.
- Your function must return its result in less than two seconds.