C Piscine

Day 04

Staff 42 pedago@42.fr

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

Contents

Chapter 1

Instructions

- The exercises are carefully laid out in order of difficulty, from easiest to hardest. An exercise is only graded if all previous ones are correct. In other words: the grading for a day stops at the first mistake.
- Be mindful of the submission procedures indicated at the start of every exercise.
- 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. Be as thorough as possible!
- Moulinette relies on a program called **norminette** to check if your files respect the Norm. An exercise containing files that do not respect the Norm will be graded 0.
- 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, it will be graded 0.
- You should not leave any additional file in your directory than those specified in the subject.

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

The forewords are entirely unrelated to the subjects and can safely be ignored.

Chapter 2

Topics

Today, you will have to learn about:

- Recursivity
- Backtracking algorithms

Chapter 3

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 4

Exercise 00 : ft_iterative_factorial

Turn-in directory : ex00/
Files to turn in: ft_iterative_factorial.c
Allowed functions: None

- Create an iterative 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_iterative_factorial(int nb);
```

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

Chapter 5

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 6

Exercise 02 : ft_iterative_power

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

- Create an iterative function that returns the value of a number raised to a given power. A 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 7

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 number raised to a given power.
- Same conditions as before.
- Here's how it should be prototyped :

```
int ft_recursive_power(int nb, int power);
```

Chapter 8

Exercise 04 : ft_fibonacci

Turn-in directory : `ex04/`
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);
```

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

Chapter 9

Exercise 05 : ft_sqrt

Turn-in directory : `ex05/`
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 10

Exercise 06 : ft_is_prime

Turn-in directory : ex06/
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 11

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 than 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 12

Exercise 08 : The Eight Queens

Turn-in directory : `ex08/`
Files to turn in: `ft_eight_queens_puzzle.c`
Allowed functions: `None`

- The eight queens puzzle is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other.
- Refresh your memories on chess rules.
- You should use recursion 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 13

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.
- You should use recursion 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
...
```

- Each solution should appear on a single line, followed by a line break
- Each solution is printed out as 8 digits: the first digit represents the queen's position in the first column, the second digit represents the queen's position in the second column, and so on. [This condensed notation is possible since a solution to the problem will have exactly one queen in each column]
- Your function must return its result in less than two seconds.