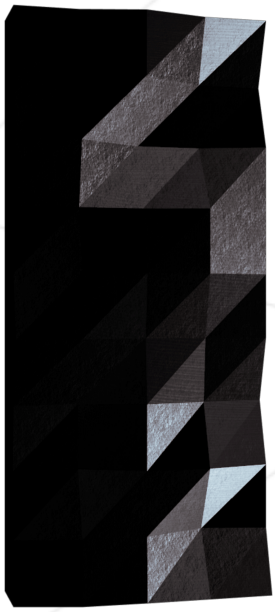

C Piscine

C 05

Summary: This document is the subject of the C 05 module of the C Piscine at 42.

Version: 7

Contents

Chapter I

Instructions

- Only this page serves as your reference, do not trust rumors.
- Watch out! This document may change before submission.
- Ensure you have the appropriate permissions on your files and directories.
- You must follow the **submission procedures** for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be evaluated by a program called **Moulinette**.
- **Moulinette** is meticulous and strict in its assessment. It is fully automated, and there is no way to negotiate with it. To avoid unpleasant surprises, be as thorough as possible.
- **Moulinette** is not open-minded. If your code does not adhere to the Norm, it won't attempt to understand it. **Moulinette** relies on a program called **norminette** to check if your files comply with the Norm. TL;DR: Submitting work that doesn't pass **norminette**'s check makes no sense.
- These exercises are arranged in order of difficulty, from easiest to hardest. We **will not** consider a successfully completed harder exercise if an easier one is not fully functional.
- Using a forbidden function is considered cheating. Cheaters receive a grade of **-42**, which is non-negotiable.
- You only need to submit a **main()** function if we specifically ask for a **program**.
- **Moulinette** compiles with the following flags: **-Wall -Wextra -Werror**, using **cc**.
- If your program does not compile, you will receive a grade of **0**.
- You **cannot** leave **any** additional file in your directory beyond those specified in the assignment.
- Have a question? Ask the peer on your right. If not, try the peer on your left.

- Your reference guide is called **Google / man / the Internet / ...**
- Check the "C Piscine" section of the forum on the intranet or the Piscine on Slack.
- Carefully examine the examples. They may contain crucial details that are not explicitly stated in the assignment...
- By Odin, by Thor! Use your brain!!!

`Norminette must be run with the -R CheckForbiddenSourceHeader flag.`
`Moulinette will use it as well.`

Chapter II

AI Instructions

Context

The C Piscine is intense. It's your first big challenge at 42 — a deep dive into problem-solving, autonomy, and community.

During this phase, your main objective is to build your foundation — through struggle, repetition, and especially **peer-learning** exchange.

In the AI era, shortcuts are easy to find. However, it's important to consider whether your AI usage is truly helping you grow — or simply getting in the way of developing real skills.

The Piscine is also a human experience — and for now, nothing can replace that. Not even AI.

For a more complete overview of our stance on AI — as a learning tool, as part of the ICT curriculum, and as a growing expectation in the job market — please refer to the dedicated FAQ available on the intranet.

Main message

- Build strong foundations without shortcuts.
- Really develop tech & power skills.
- Experience real peer-learning, start learning how to learn and solve new problems.
- The learning journey is more important than the result.
- Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.
- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

Comments and example:

- Yes, we know AI exists — and yes, it can solve your projects. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

✓ Good practice:

I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

✗ Bad practice:

I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

Chapter III

Foreword

Here is an excerpt from the lyrics of 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 has nothing to do with the *Harry Potter* saga, which is a shame, because your exercises won't be completed by *magic*!

Chapter IV

Exercise 00 : ft_iterative_factorial

	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 a number. This number should be the result of a factorial operation based on the given parameter.
- If the argument is not valid, the function should return 0.
- Overflows do not need to be handled; the function's return value will be undefined in such cases.
- The function should be prototyped as follows:

```
int ft_iterative_factorial(int nb);
```

Chapter V

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 given parameter.
- If the argument is not valid, the function should return 0.
- Overflows do not need to be handled; the function's return value will be undefined in such cases.
- The function should be prototyped as follows:

```
int ft_recursive_factorial(int nb);
```

Chapter VI

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 iterative function that returns the result of raising a number to a given power.
- If the power is less than 0, the function should return 0.
- Overflows do not need to be handled.
- By definition, 0 raised to the power of 0 should return 1.
- The function should be prototyped as follows:

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

Chapter VII

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 result of raising a number to a given power.
- If the power is less than 0, the function should return 0.
- Overflows do not need to be handled; the function's return value will be undefined in such cases.
- By definition, 0 raised to the power of 0 should return 1.
- The function should be prototyped as follows:

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

Chapter VIII

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, with the first element at index 0.
  The Fibonacci sequence will be considered to start as follows: 0, 1, 1, 2.
- Overflows do not need to be handled; the function's return value will be undefined in such cases.
- The function should be prototyped as follows:

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

- `ft_fibonacci` must be implemented recursively.
- If `index` is less than 0, the function should return -1.

Chapter IX

Exercise 05 : ft_sqrt

	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 given number (if it exists), or 0 if the square root is an irrational number.
- The function should be prototyped as follows:

```
int ft_sqrt(int nb);
```

Chapter X

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 given number is a prime number and 0 if it is not.
- The function should be prototyped as follows:

```
int ft_is_prime(int nb);
```

`0 and 1 are not prime numbers.`

Chapter XI

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 than or equal to the given number.
- The function should be prototyped as follows:

```
int ft_find_next_prime(int nb);
```

Chapter XII

Exercise 08 : The Ten Queens

	Exercise 08
The Ten Queens	
Turn-in directory: $ex08/$	
Files to turn in: `ft_ten_queens_puzzle.c`	
Allowed functions: `write`	

- Create a function that displays all possible placements of ten queens on a 10×10 chessboard, ensuring that no two queens can attack each other in a single move. The function should return the total number of valid solutions.
- Recursion is required to solve this problem.
- The function should be prototyped as follows:

```
int ft_ten_queens_puzzle(void);
```

- Output format:

```
$>./a.out | cat -e
0257948136$
0258693147$
...
4605713829$
4609582731$
...
9742051863$
$>
```

- The sequence is read from left to right, where:
  - The first digit represents the row position of the queen in the first column (index starting at 0).
  - The Nth digit represents the row position of the queen in the Nth column.
- The function should return the total number of valid solutions found.

Chapter XIII

Submission and peer-evaluation

Submit your assignment to your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Make sure to double-check the names of your files to ensure they are correct.

`You must submit only the files required by the project specifications.`