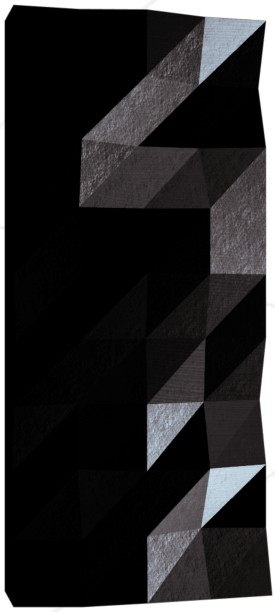

C Piscine

C 03

Summary: This document is the subject for the C 03 module of the C Piscine @ 42.

Version: 5

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

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

The earliest known mention of the game Rock-Paper-Scissors (RPS) appears in the book *Wuzazu*, written by the Chinese Ming-dynasty author Xie Zhaozhi. He noted that the game dates back to the Chinese Han dynasty (206 BC – 220 AD). In *Wuzazu*, the game was called *shoushiling*.

Li Rihua's book *Note of Liuyanzhai* also references this game, calling it *shoushiling*, *huozhitou*, or *huoquan*.

Throughout Japanese history, there are frequent mentions of *sansukumi-ken*, meaning "ken" (fist) games with a three-way *sukumi* (deadlock). In these games, A defeats B, B defeats C, and C defeats A. The games originated in China before being introduced to Japan, where they became widely popular.

By the early 20th century, Rock-Paper-Scissors had spread beyond Asia, largely due to increased Japanese interactions with the West. Its English name is derived from a translation of the three Japanese hand gestures representing rock, paper, and scissors. Elsewhere in Asia, the open-palm gesture typically represents "cloth" rather than "paper". The depiction of scissors also follows the Japanese style.

In 1927, *La Vie au Patronage*, a French children's magazine, described the game in detail, referring to it as a *jeu japonais* ("Japanese game"). Its French name, *Chi-fou-mi*, is based on the Old Japanese words for "one, two, three" (*hi, fu, mi*).

A 1932 *New York Times* article on Tokyo's rush hour explained the game's rules for American readers, suggesting that it was not yet widely known in the U.S. at the time. The 1933 edition of *Compton's Pictured Encyclopedia* described it as a common method for children to settle disputes in Japan, calling it "John Kem Po." The article even noted, "This is such a good way of deciding an argument that American boys and girls might like to practice it too."

Chapter IV

Exercise 00 : ft_strcmp

	Exercise 00
ft_strcmp	
Turn-in directory: $ex00/$	
Files to turn in: `ft_strcmp.c`	
Allowed functions: `None`	

- Reproduce the behavior of the function `strcmp` (man strcmp).
- The function should be prototyped as follows:

```
int        ft_strcmp(char *s1, char *s2);
```

Chapter V

Exercise 01 : ft_strncmp

	Exercise 01
ft_strncmp	
Turn-in directory: *ex*01/	
Files to turn in: `ft_strncmp.c`	
Allowed functions: `None`	

- Reproduce the behavior of the function `strncmp` (man strncmp).
- The function should be prototyped as follows:

```
int        ft_strncmp(char *s1, char *s2, unsigned int n);
```

Chapter VI

Exercise 02 : ft_strcat

	Exercise 02
ft_strcat	
Turn-in directory: *ex*02/	
Files to turn in: `ft_strcat.c`	
Allowed functions: `None`	

- Reproduce the behavior of the function `strcat` (man strcat).
- The function should be prototyped as follows:

```
char *ft_strcat(char *dest, char *src);
```

Chapter VII

Exercise 03 : ft_strncat

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

- Reproduce the behavior of the function `strncat` (man strncat).
- The function should be prototyped as follows:

```
char *ft_strncat(char *dest, char *src, unsigned int nb);
```

Chapter VIII

Exercise 04 : ft_strstr

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

- Reproduce the behavior of the function **strstr** (man strstr).
- The function should be prototyped as follows:

```
char        *ft_strstr(char *str, char *to_find);
```

Chapter IX

Exercise 05 : ft_strlcat

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

- Reproduce the behavior of the function `strlcat` (man strlcat).
- The function should be prototyped as follows:

```
unsigned int ft_strlcat(char *dest, char *src, unsigned int size);
```

Chapter X

Submission and peer-evaluation

Submit your assignment in 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 explicitly requested by the project requirements.`