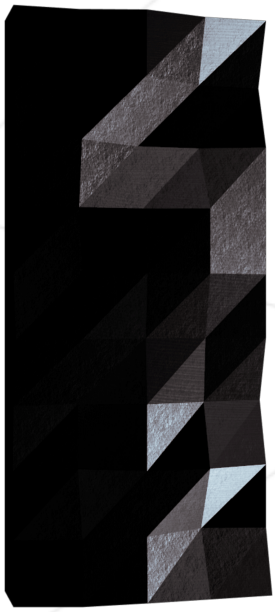

C Piscine

C 07

Staff 42 pedago@42.fr

Summary: This document is the subject for the module C 07 of the C Piscine @ 42.

Version: 6

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 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

Morty: Rick!

Rick: Uhp-uhp-uhp! Morty, keep your hands off your ding-dong! It's the only way we can speak freely. Look around you, Morty. Do you really think this wuh-world is real? You'd have to be an idiot not to notice all the sloppy details. Look, that guy's putting a bun between two hot dogs.

Morty: I dunno, Rick, I mean, I've seen people do that before.

Rick: Well, look at that old lady. She's-she's walking a cat on a leash.

Morty: Uh, Mrs. Spencer does that all the time, Rick.

Rick: Look, I-I-I don't want to hear about Mrs. Spencer, Morty! She's an idiot! All right, all right, there. Wh-what about that, Morty?

Morty: Okay, okay, you got me on that one.

Rick: Oh, really, Morty? Are you sure you haven't seen that somewhere in real life before?

Morty: No, no, I haven't seen that. I mean, why would a Pop-Tart want to live inside a toaster, Rick? I mean, th-that would be like the scariest place for them to live. Y'know what I mean?

Rick: You're missing the point, Morty. Why would he drive a smaller toaster with wheels? I mean, does your car look like a smaller version of your house? No.

Morty: So, why are they doing this? W-what do they want?

Rick: Well, that would be obvious to you, Morty, if you'd been paying attention.
[an ambulance drives past Rick and Morty and stops; open back doors]
Paramedic: We got the President of the United States in here! We need 10cc of concentrated dark matter, stat, or he'll die!

Morty: Concentrated dark matter? They were asking about that in class.

Rick: Yeah, it’s a special fuel I invented to travel through space faster than anybody else. These Zigerions are always trying to scam me out of my secrets, but they made a big mistake this time, Morty. They dragged you into this. Now they’re gonna pay!

Morty: What do you- w-w-what are we gonna do?

Rick: We’re gonna scam the scammers, Morty. And we’re gonna take ’em for everything they’ve got.

The following exercises will be easier to complete if you are a fan of "Rick and Morty"

Chapter IV

Exercise 00 : ft_strdup

	Exercise 00
ft_strdup	
Turn-in directory: $ex00/$	
Files to turn in: `ft_strdup.c`	
Allowed functions: `malloc`	

- Reproduce the behavior of the function `strdup` (man strdup).
- Here's how it should be prototyped :

```
char    *ft_strdup(char *src);
```

Chapter V

Exercise 01 : ft_range

	Exercise 01
ft_range	
Turn-in directory: *ex*01/	
Files to turn in: `ft_range.c`	
Allowed functions: `malloc`	

- Create a function `ft_range` which returns an array of `ints`. This `int` array should contain all values between `min` and `max`.
- `Min` included - `max` excluded.
- Here is how it should be prototyped :

```
int    *ft_range(int min, int max);
```

- If the value of `min` is greater than or equal to `max`, a NULL pointer should be returned.

Chapter VI

Exercise 02 : ft_ultimate_range

	Exercise 02
ft_ultimate_range	
Turn-in directory: *ex*02/	
Files to turn in: `ft_ultimate_range.c`	
Allowed functions: `malloc`	

- Create a function `ft_ultimate_range` which allocates and assigns an array of `ints`. This `int` array should contain all values between `min` and `max`.
- `Min` included - `max` excluded.
- Here is how it should be prototyped :

```
int    ft_ultimate_range(int **range, int min, int max);
```

- The size of `range` should be returned (or -1 on error).
- If the value of `min` is greater or equal to `max`'s value, `range` will point to NULL and it should return 0.

Chapter VII

Exercice 03 : ft_strjoin

	Exercise 03
ft_strjoin	
Turn-in directory: $ex03/$	
Files to turn in: `ft_strjoin.c`	
Allowed functions: `malloc`	

- Write a function that concatenates all the strings pointed to by `strs`, separated by `sep`.
- `size` is the number of strings in `strs`.
- If `size` is 0, you must return an empty string that can be freed using `free()`.
- Here is how it should be prototyped:

```
char *ft_strjoin(int size, char **strs, char *sep);
```

Chapter VIII

Exercise 04 : ft_convert_base

	Exercise 04
ft_convert_base	
Turn-in directory: *ex*04/	
Files to turn in: `ft_convert_base.c, ft_convert_base2.c`	
Allowed functions: `malloc, free`	

- Create a function that returns the result of converting the string `nbr` from base `base_from` to base `base_to`.
- `nbr`, `base_from`, and `base_to` may be read-only.
- `nbr` will follow the same rules as `ft_atoi_base` (from another module). Beware of the characters '+', '-', and whitespaces.
- The number represented by `nbr` must fit inside an `int`.
- If a base is invalid, `NULL` should be returned.
- The returned number must be prefixed only by a single and unique '-' if necessary; no whitespaces, no '+'.
- Here is how it should be prototyped:

```
char *ft_convert_base(char *nbr, char *base_from, char *base_to);
```

Chapter IX

Exercise 05 : ft_split

	Exercise 05
ft_split	
Turn-in directory: *ex*05/	
Files to turn in: `ft_split.c`	
Allowed functions: `malloc`	

- Create a function that splits a string of characters based on an additional string of characters.
- You'll have to use each character from the string `charset` as a separator.
- The function should return an array where each element of the array contains the address of a string, wrapped between two separators. The last element of the array should be `NULL` to indicate the end of the array.
- There cannot be any empty strings in your array. Draw your own conclusions accordingly.
- The string given as an argument won't be modifiable.
- Here's how it should be prototyped:

```
char **ft_split(char *str, char *charset);
```

Chapter X

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 filenames to ensure they are correct.

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