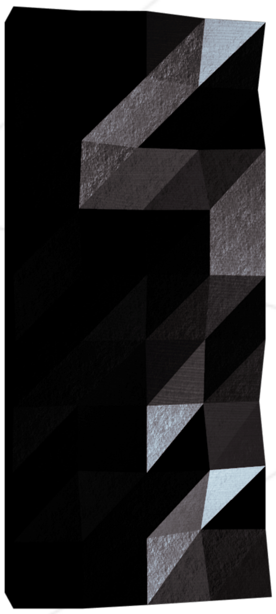

C Piscine

C 09

Summary: This document is the subject for the module C 09 of the C Piscine @ 42.

Version: 5.0

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

Dialog from the movie `The Big Lebowski`:

```
The Dude: Walter, ya know, it's Smokey, so his toe slipped over the line a little,
  big deal. It's just a game, man.
Walter Sobchak: Dude, this is a league game, this determines who enters the next
  round robin. Am I wrong? Am I wrong?
Smokey: Yeah, but I wasn't over. Gimme the marker Dude, I'm marking it 8.
Walter Sobchak: [pulls out a gun] Smokey, my friend, you are entering a world
  of pain.
The Dude: Walter...
Walter Sobchak: You mark that frame an 8, and you're entering a world of pain.
Smokey: I'm not...
Walter Sobchak: A world of pain.
Smokey: Dude, he's your partner...
Walter Sobchak: [shouting] Has the whole world gone crazy? Am I the only one
  around here who gives a shit about the rules? Mark it zero!
The Dude: They're calling the cops, put the piece away.
Walter Sobchak: Mark it zero!
[points gun in Smokey's face]
The Dude: Walter...
Walter Sobchak: [shouting] You think I'm fucking around here? Mark it zero!
Smokey: All right, it's fucking zero. Are you happy, you crazy fuck?
Walter Sobchak: ...It's a league game, Smokey.
```

Chapter IV

Exercise 00 : libft

	Exercise 00
libft	
Turn-in directory: *ex*00/	
Files to turn in: `libft_creator.sh`, `ft_putchar.c`, `ft_swap.c`, `ft_putstr.c`, `ft_strlen.c`, `ft_strcmp.c`	
Allowed functions: `write`	

- Create your `ft` library. It will be called `libft.a`.
- A shell script called `libft_creator.sh` will compile the source files appropriately and will create your library.
- This library should contain all of the following functions :

```
void    ft_putchar(char c);
void    ft_swap(int *a, int *b);
void    ft_putstr(char *str);
int     ft_strlen(char *str);
int     ft_strcmp(char *s1, char *s2);
```

- We'll launch the following command-line :

```
sh libft_creator.sh
```

Chapter V

Exercise 01 : Makefile

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

- Create the `Makefile` that will compile a library `libft.a`.
- Your Makefile should print all the commands it's running.
- Your Makefile should not run any unnecessary commands.
- The `Makefile` will get its source files from the "srcs" directory.
- These files will be: `ft_putchar.c`, `ft_swap.c`, `ft_putstr.c`, `ft_strlen.c`, `ft_strcmp.c`.
- The `Makefile` will get its header files from the "includes" directory.
- These files will be: `ft.h`.
- It should compile the .c files with `cc` and with `-Wall -Wextra -Werror` flags in that order.
- The lib should be at the root of the exercise.
- .o files should be near their corresponding .c files.
- The `Makefile` should also implement the following rules: `clean`, `fclean`, `re`, `all`, and of course `libft.a`.
- Running just `make` should be equivalent to `make all`.
- The `all` rule should be equivalent to `make libft.a`.
- The `clean` rule should remove all the temporary generated files.

- The `fclean` rule should be like a `make clean`, plus removing all the binaries generated with `make all`.
- The `re` rule should be like a `make fclean` followed by `make all`.
- Your Makefile should not compile any file unnecessarily.
- We'll only fetch your Makefile and test it with our files.

```
Watch out for wildcards!
```

Chapter VI

Exercise 02 : ft_split

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

- Create a function that splits a string, using each character from the `charset` string as a separator.
- You will need to use each character from the `charset` string individually as a separator.
- The function should return an array where each element contains the address of a substring wrapped between two separators. The last element of the array should be `NULL` to indicate the end of the array.
- There should be no empty strings in your array. Draw your conclusions accordingly.
- The string provided as an argument cannot be modified.
- Here's the function prototype:

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

Chapter VII

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.`