



C Piscine

C 09

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

# Contents

<b>I</b>	<b>Instructions</b>	<b>2</b>
<b>II</b>	<b>Foreword</b>	<b>4</b>
<b>III</b>	<b>Exercise 00 : libft</b>	<b>5</b>
<b>IV</b>	<b>Exercise 01 : Makefile</b>	<b>6</b>
<b>V</b>	<b>Exercise 02 : ft_split</b>	<b>8</b>

# Chapter I

## Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- 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. So if you want to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called `norminette` to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass `norminette`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We **will not** take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get **-42**, and this grade is non-negotiable.
- 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, you'll get 0.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.

- Your reference guide is called `Google / man / the Internet / ....`
- Check out the "C Piscine" part of the forum on the intranet, or the slack Piscine.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor ! Use your brain !!!



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

# Chapter II

## 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 III

## Exercise 00 : libft

	Exercise 00
libft	
Turn-in directory : <i>ex00/</i>	
Files to turn in : <code>libft_creator.sh</code> , <code>ft_putchar.c</code> , <code>ft_swap.c</code> , <code>ft_putstr.c</code> , <code>ft_strlen.c</code> , <code>ft_strcmp.c</code>	
Allowed functions : <code>write</code>	

- Create your `ft` library. It'll be called `libft.a`.
- A shell script called `libft_creator.sh` will compile 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 IV

## Exercise 01 : Makefile

	Exercise 01
Makefile	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <b>Makefile</b>	
Allowed functions : <b>None</b>	

- Create the **Makefile** that'll compile a library **libft.a**.
- Your makefile should print all the command it's running.
- Your makefile should not run any unnecessary command.
- The **Makefile** will get its source files from the "srcs" directory.
- Those files will be: `ft_putchar.c`, `ft_swap.c`, `ft_putstr.c`, `ft_strlen.c`, `ft_stremp.c`
- The **Makefile** will get its header files from the "includes" directory.
- Those files will be: `ft.h`
- It should compile the `.c` files with `gcc` 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 `.c` file.
- The **Makefile** should also implement the following rules: **clean**, **fclean**, **re**, **all** and of course **libft.a**.
- Running just **make** should be equal to **make all**
- The rule **all** should be equal to **make libft.a**.
- The rule **clean** should remove all the temporary generated files.

- The rule `fclean` should be like a `make clean` plus all the binary made with `make all`.
- The rule `re` should be like a `make fclean` followed by `make all`.
- Your makefile should not compile any file for nothing.
- We'll only fetch your Makefile and test it with our files.




Watch out for wildcards!



# Chapter V

## Exercise 02 : ft\_split

	Exercise 02
ft_split	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>ft_split.c</b>	
Allowed functions : <b>malloc</b>	

- Create a function that splits a string of character depending on another string of characters.
- You'll have to use each character from the string **charset** as a separator.
- The function returns an array where each box contains the address of a string wrapped between two separators. The last element of that array should equal to 0 to indicate the end of the array.
- There cannot be any empty strings in your array. Draw your conclusions accordingly.
- The string given as argument won't be modifiable.
- Here's how it should be prototyped :

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