

# Contents

I	Instructions	2
II	Foreword	4
III	Exercise 00 : ft_print_alphabet	5
IV	Exercise 01 : ft_print_reverse_alphabet	6
V	Exercise 02 : ft_print_numbers	7
VI	Exercise 03: ft_is_negative	8
VII	Exercise 04 : ft_print_comb	9
VIII	Exercise 05 : ft_print_comb2	10
IX	Exercise 06 : ft_putnbr	11
X	Exercise 07 : ft_print_combn	12

# Chapter I

## Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up to an hour 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 **Norminator** to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass **Norminator**'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.
- If `ft_putchar()` is an authorized function, we will compile your code with our `ft_putchar.c`.
- 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.
- 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 !!!



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

# Chapter II

## Foreword

Cod liver oil is a nutritional supplement derived from liver of cod fish (Gadidae).

As with most fish oils, it has high levels of the omega-3 fatty acids, eicosapentaenoic acid (EPA) and docosahexaenoic acid (DHA). Cod liver oil also contains vitamin A and vitamin D.


It has historically been taken because of its vitamin A and vitamin D content.

It was once commonly given to children, because vitamin D has been shown to prevent rickets and other symptoms of vitamin D deficiency.

Contrary to Cod liver oil, C is good, eat some!

# Chapter III

## Exercise 00 : ft\_print\_alphabet


	Exercise : 00
ft_print_alphabet	
Turn-in directory : <i>ex00/</i>	
Files to turn in : <code>ft_print_alphabet.c</code>	
Allowed functions : <code>ft_putchar</code>	
Remarks : n/a	

- Create a function that displays the alphabet in lowercase, on a single line, by ascending order, starting from the letter 'a'.
- Here's how it should be prototyped :

```
void ft_print_alphabet(void);
```

## Chapter IV

### Exercise 01 : ft\_print\_reverse\_alphabet


	Exercice : 01
	ft_print_reverse_alphabet
	Turn-in directory : <i>ex01/</i>
	Files to turn in : <code>ft_print_reverse_alphabet.c</code>
	Allowed functions : <code>ft_putchar</code>
	Remarks : n/a

- Create a function that displays the alphabet in lowercase, on a single line, by descending order, starting from the letter 'z'.
- Here's how it should be prototyped :

```
void ft_print_reverse_alphabet(void);
```

# Chapter V

## Exercise 02 : ft\_print\_numbers


	Exercise : 02
	ft_print_numbers
Turn-in directory : <i>ex02/</i>	
Files to turn in : <code>ft_print_numbers.c</code>	
Allowed functions : <code>ft_putchar</code>	
Remarks : n/a	

- Create a function that displays all digits, on a single line, by ascending order.
- Here's how it should be prototyped :

```
void ft_print_numbers(void);
```

# Chapter VI

## Exercise 03: ft\_is\_negative

	Exercice : 03
	ft_is_negative
Turn-in directory : <i>ex03/</i>	
Files to turn in : <b>ft_is_negative.c</b>	
Allowed functions : <b>ft_putchar</b>	
Remarks : n/a	


- Create a function that displays 'N' or 'P' depending on the integer's sign entered as a parameter. If `n` is negative, display 'N'. If `n` is positive or null, display 'P'.
- Here's how it should be prototyped :

```
void ft_is_negative(int n);
```



# Chapter VII

## Exercise 04 : ft\_print\_comb

	Exercice : 04
	ft_print_comb
	Turn-in directory : <i>ex04/</i>
	Files to turn in : <b>ft_print_comb.c</b>
	Allowed functions : <b>ft_putchar</b>
	Remarks : n/a

- Create a function that displays all different combinations of three different digits in ascending order, listed by ascending order - yes, repetition is voluntary.
- Here's the intended output :


```
$>./a.out | cat -e
012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789$>
```

- 987 isn't there because 789 already is.
- 999 isn't there because the digit 9 is present more than once.
- Here's how it should be prototyped :

```
void ft_print_comb(void);
```

# Chapter VIII

## Exercise 05 : ft\_print\_comb2

	Exercise : 05
	ft_print_comb2
	Turn-in directory : <i>ex05/</i>
	Files to turn in : <b>ft_print_comb2.c</b>
	Allowed functions : <b>ft_putchar</b>
	Remarks : n/a

- Create a function that displays all different combination of two digits between 00 and 99, listed by ascending order.
- Here's the expected output :


```
$>./a.out | cat -e
00 01, 00 02, 00 03, 00 04, 00 05, ..., 00 99, 01 02, ..., 97 99, 98 99$>
```

- Here's how it should be prototyped :

```
void ft_print_comb2(void);
```

# Chapter IX

## Exercise 06 : ft\_putnbr

	Exercice : 06
ft_putnbr	
Turn-in directory : <i>ex06/</i>	
Files to turn in : <b>ft_putnbr.c</b>	
Allowed functions : <b>ft_putchar</b>	
Remarks : n/a	


- Create a function that displays the number entered as a parameter. The function has to be able to display all possible values within an `int` type variable.
- Here's how it should be prototyped :

```
void ft_putnbr(int nb);
```

- For example:
  - `ft_putnbr(42)` displays "42".

# Chapter X

## Exercise 07 : ft\_print\_combn

	Exercice : 07
	ft_print_combn
	Turn-in directory : <i>ex07/</i>
	Files to turn in : <b>ft_print_combn.c</b>
	Allowed functions : <b>ft_putchar</b>
	Remarks : n/a

- Create a function that displays all different combinations of **n** numbers by ascending order.
- **n** will be so that :  $0 < n < 10$ .
- If **n** = 2, here's the expected output :

```
$>./a.out | cat -e
01, 02, 03, ..., 09, 12, ..., 79, 89$>
```

- Here's how it should be prototyped :

```
void ft_print_combn(int n);
```