Contents

I	Instructions	2
II	Foreword	4
III	Exercise 00 : libft	5
IV	Exercise 01 : ft_print_program_name	6
V	Exercise 02 : ft_print_params	7
VI	Exercise 03 : ft_rev_params	8
VII	Exercise 04 : ft_sort_params	9

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 every exercise.
- 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 your right. Otherwise, try your peer on your 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

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

	Exercice : 00
libft	
Turn-in directory : $ex00/$	
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`	
Remarks : `n/a`	

- 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
```

Don't hesitate to add other useful functions... ;-)

Chapter IV

Exercise 01 : ft_print_program_name

	Exercice : 01
ft_print_program_name	
Turn-in directory : $ex01/$	
Files to turn in : `ft_print_program_name.c`	
Allowed functions : `ft_putchar`	
Remarks : `n/a`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its own name.
- Example :

```
$>./a.out
./a.out
$>
```

Chapter V

Exercise 02 : ft_print_params

	Exercice : 02
ft_print_params	
Turn-in directory : $ex02/$	
Files to turn in : `ft_print_params.c`	
Allowed functions : `ft_putchar`	
Remarks : `n/a`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its given arguments.
- Example :

```
$>./a.out test1 test2 test3
test1
test2
test3
$>
```

Chapter VI

Exercise 03 : ft_rev_params

	Exercice : 03
ft_rev_params	
Turn-in directory : $ex03/$	
Files to turn in : `ft_rev_params.c`	
Allowed functions : `ft_putchar`	
Remarks : `n/a`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its given arguments in reverse order.
- It should display all arguments, except for `argv[0]`.
- All arguments have to have their own line.

Chapter VII

Exercise 04 : ft_sort_params

	Exercice : 04
ft_sort_params	
Turn-in directory : $ex04/$	
Files to turn in : `ft_sort_params.c`	
Allowed functions : `ft_putchar`	
Remarks : `n/a`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its given arguments sorted by ascii order.
- It should display all arguments, except for `argv[0]`.
- All arguments have to have their own line.