

C Piscine C 02

Summary: This document is the subject for the C 02 module of the C Piscine @ 42.

Version: 5.2

# Contents

1	mstructions	4
II	Foreword	4
III	Exercise 00 : ft_strcpy	5
IV	Exercise 01 : ft_strncpy	6
$\mathbf{V}$	Exercise 02 : ft_str_is_alpha	7
VI	Exercise 03 : ft_str_is_numeric	8
VII	Exercise 04 : ft_str_is_lowercase	9
VIII	Exercise 05 : ft_str_is_uppercase	10
IX	Exercise 06 : ft_str_is_printable	11
$\mathbf{X}$	Exercise 07 : ft_strupcase	12
XI	Exercise 08 : ft_strlowcase	13
XII	Exercise 09 : ft_strcapitalize	14
XIII	Exercise 10 : ft_strlcpy	15
XIV	Exercise 11 : ft_putstr_non_printable	16
XV	Exercise 12 : ft_print_memory	17
XVI	Submission and peer-evaluation	19

#### Chapter I

#### Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up 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 cc.
- If your program doesn't compile, you'll get 0.
- You <u>cannot</u> leave <u>any</u> 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.

- $\bullet$  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  $\mbox{-R CheckForbiddenSourceHeader}$  flag. Moulinette will use it too.

#### Chapter II

#### Foreword

Here is a discuss extract from the Silicon Valley serie:

- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emac.
- Oh, God, help us! Okay, uh you know what? I just don't think this is going to work. I'm so sorry. Uh, I mean like, what, we're going to bring kids into this world with that over their heads? That's not really fair to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no way I'm going to be with someone who uses spaces over tabs.
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! (DOOR SLAMS) (BANGING)

. ,

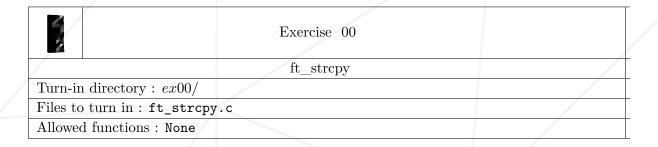
#### (RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time. I'm okay, though.
- See you around, Richard.
- Just making a point.

Hopefully, you are not forced to use emacs and your space bar to complete the following exercices.

# Chapter III

Exercise 00: ft\_strcpy

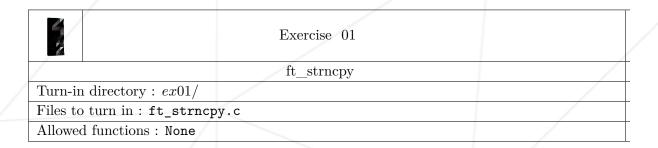


- Reproduce the behavior of the function strcpy (man strcpy).
- Here's how it should be prototyped :

char \*ft\_strcpy(char \*dest, char \*src);

# Chapter IV

# Exercise 01: ft\_strncpy

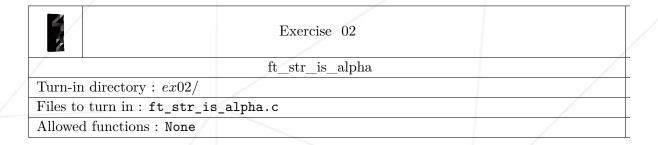


- Reproduce the behavior of the function strncpy (man strncpy).
- Here's how it should be prototyped :

char \*ft\_strncpy(char \*dest, char \*src, unsigned int n);

## Chapter V

Exercise 02: ft\_str\_is\_alpha

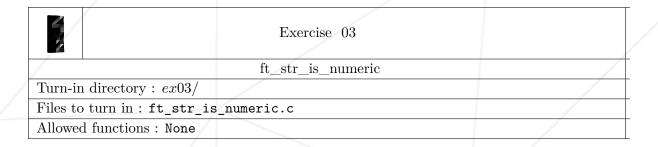


- Create a function that returns 1 if the string given as a parameter contains only alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped:

int ft\_str\_is\_alpha(char \*str);

## Chapter VI

Exercise 03: ft\_str\_is\_numeric

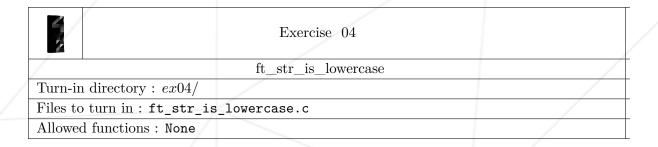


- Create a function that returns 1 if the string given as a parameter contains only digits, and 0 if it contains any other character.
- Here's how it should be prototyped :

int ft\_str\_is\_numeric(char \*str);

#### Chapter VII

# Exercise 04: ft\_str\_is\_lowercase

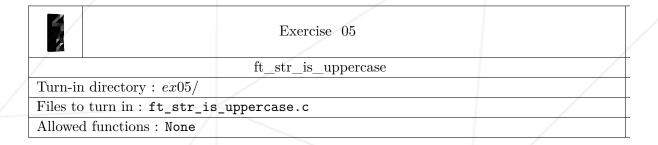


- Create a function that returns 1 if the string given as a parameter contains only lowercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

int ft\_str\_is\_lowercase(char \*str);

# Chapter VIII

Exercise  $05: ft\_str\_is\_uppercase$ 

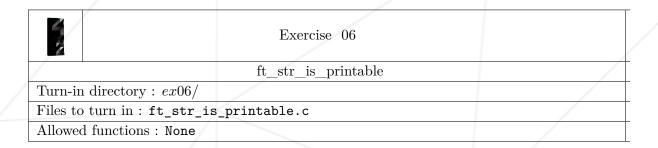


- Create a function that returns 1 if the string given as a parameter contains only uppercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped:

int ft\_str\_is\_uppercase(char \*str);

#### Chapter IX

Exercise 06: ft\_str\_is\_printable

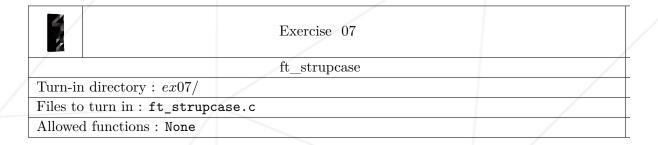


- Create a function that returns 1 if the string given as a parameter contains only printable characters, and 0 if it contains any other character.
- Here's how it should be prototyped:

int ft\_str\_is\_printable(char \*str);

# Chapter X

# Exercise 07: ft\_strupcase



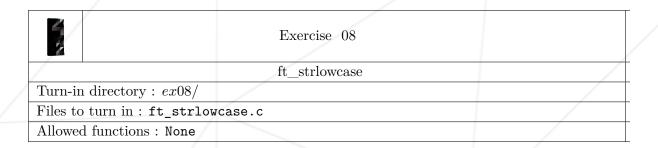
- Create a function that transforms every letter to uppercase.
- Here's how it should be prototyped :

char \*ft\_strupcase(char \*str);

• It should return str.

# Chapter XI

# Exercise 08: ft\_strlowcase



- Create a function that transforms every letter to lowercase.
- $\bullet$  Here's how it should be prototyped :

char \*ft\_strlowcase(char \*str);

• It should return str.

#### Chapter XII

# Exercise 09: ft\_strcapitalize

	Exercise 09	
/	${\it ft\_strcapitalize}$	
Turn-in directory : $ex09/$	/	
Files to turn in : ft_str		
Allowed functions : None		

- Create a function that capitalizes the first letter of each word and transforms all other letters to lowercase.
- A word is a string of alphanumeric characters.
- Here's how it should be prototyped :

```
char *ft_strcapitalize(char *str);
```

- It should return str.
- For example:

```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

• Becomes:

Salut, Comment Tu Vas ? 42mots Quarante-Deux; Cinquante+Et+Un

# Chapter XIII

Exercise 10: ft\_strlcpy

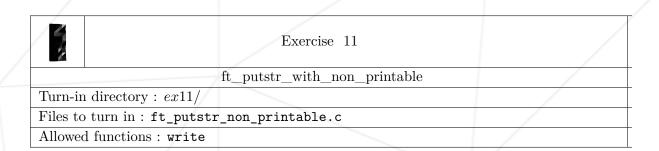
	Exercise 10	
/	ft_strlcpy	
Turn-in directory : $ex10/$		
Files to turn in : ft_strlcp		
Allowed functions : None		

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

unsigned int ft\_strlcpy(char \*dest, char \*src, unsigned int size);

## Chapter XIV

# Exercise 11: ft\_putstr\_non\_printable



- Create a function that displays a string of characters onscreen. If this string contains characters that aren't printable, they'll have to be displayed in the shape of hexadecimals (lowercase), preceded by a "backslash".
- For example:

Coucou\ntu vas bien ?

• The function should display:

Coucou\Oatu vas bien ?

• Here's how it should be prototyped:

void ft\_putstr\_non\_printable(char \*str);

#### Chapter XV

#### Exercise 12: ft\_print\_memory

	Exercise 12	
/	ft_print_memory	
Turn-in directory : $ex12/$		
Files to turn in : ft_prin		
Allowed functions: write		

- Create a function that displays the memory area onscreen.
- The display of this memory area should be split into three "columns" separated by a space:
  - The hexadecimal address of the first line's first character followed by a ':'.
  - The content in hexadecimal with a space each 2 characters and should be padded with spaces if needed (see the example below).
  - The content in printable characters.
- If a character is non-printable, it'll be replaced by a dot.
- Each line should handle sixteen characters.
- If size is equal to 0, nothing should be displayed.

C Piscine

• Example:

```
$> ./ft_print_memory
00000010a161f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin
000000010a161f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo
000000010a161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
00000010a161f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.
000000010a161f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..
000000010a161f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000 ..lol.lol..
$> ./ft_print_memory | cat -te
0000000107ff9f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin$
000000107ff9f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo$
0000000107ff9f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on $
0000000107ff9f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.$
0000000107ff9f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..$
0000000107ff9f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000 ..lol.lol. .$
$>
```

• Here's how it should be prototyped:

```
void *ft_print_memory(void *addr, unsigned int size);
```

• It should return addr.

# Chapter XVI

## Submission and peer-evaluation

Turn in your assignment in your Git repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.



You need to return only the files requested by the subject of this project.