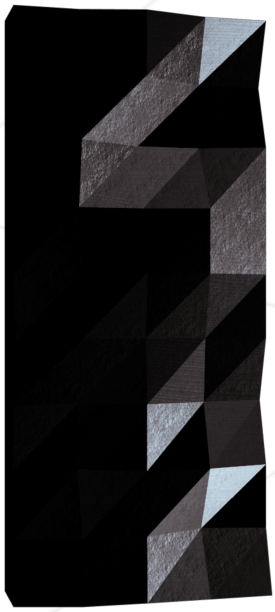


C Piscine

C 00

Summary: THIS document is the subject for the C 00 module of the C Piscine @ 42.

Contents

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 `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

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

Exercice 00 : ft_putchar

	Exercice 00
ft_putchar	
Turn-in directory : $ex00/$	
Files to turn in : `ft_putchar.c`	
Allowed functions : `write`	

- Write a function that displays the character passed as a parameter.
- It will be prototyped as follows :

```
void ft_putchar(char c);
```

To display the character, you must use the texttt write function as follows.

```
write(1, &c, 1);
```

Chapter IV

Exercise 01 : ft_print_alphabet

	Exercise 01
ft_print_alphabet	
Turn-in directory : $ex01/$	
Files to turn in : `ft_print_alphabet.c`	
Allowed functions : `write`	

- 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 V

Exercise 02 : ft_print_reverse_alphabet

	Exercise 02
ft_print_reverse_alphabet	
Turn-in directory : $ex02/$	
Files to turn in : `ft_print_reverse_alphabet.c`	
Allowed functions : `write`	

- 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 VI

Exercise 03 : ft_print_numbers

	Exercise 03
ft_print_numbers	
Turn-in directory : $ex03/$	
Files to turn in : `ft_print_numbers.c`	
Allowed functions : `write`	

- 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 VII

Exercise 04 : ft_is_negative

	Exercise 04
ft_is_negative	
Turn-in directory : $ex04/$	
Files to turn in : `ft_is_negative.c`	
Allowed functions : `write`	

- 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 VIII

Exercise 05 : ft_print_comb

	Exercise 05
ft_print_comb	
Turn-in directory : $ex05/$	
Files to turn in : `ft_print_comb.c`	
Allowed functions : `write`	

- 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 IX

Exercise 06 : ft_print_comb2

	Exercise 06
ft_print_comb2	
Turn-in directory : $ex06/$	
Files to turn in : `ft_print_comb2.c`	
Allowed functions : `write`	

- 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 X

Exercise 07 : ft_putnbr

	Exercise 07
ft_putnbr	
Turn-in directory : *ex*07/	
Files to turn in : `ft_putnbr.c`	
Allowed functions : `write`	

- 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 XI

Exercise 08 : ft_print_combn

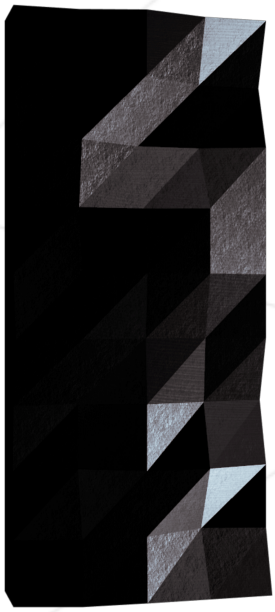
	Exercise 08
ft_print_combn	
Turn-in directory : *ex*08/	
Files to turn in : `ft_print_combn.c`	
Allowed functions : `write`	

- 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);
```



C Piscine

C 01

Summary: This document is the subject for the module C 01 of the C Piscine @ 42.

Contents

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 `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

Vincent: And you know what they call a... a... a Quarter Pounder with Cheese in Paris?

Jules: They don't call it a Quarter Pounder with cheese?

Vincent: No man, they got the metric system. They wouldn't know what the fuck a Quarter Pounder is.

Jules: Then what do they call it?

Vincent: They call it a Royale with cheese.

Jules: A Royale with cheese. What do they call a Big Mac?

Vincent: Well, a Big Mac's a Big Mac, but they call it le Big-Mac.

Jules: Le Big-Mac. Ha ha ha ha. What do they call a Whopper?

Vincent: I dunno, I didn't go into Burger King.

At least one of the following exercices has nothing to do you with a Royale with cheese.

Chapter III

Exercise 00 : ft_ft

	Exercise 00
ft_ft	
Turn-in directory : *ex*00/	
Files to turn in : `ft_ft.c`	
Allowed functions : `None`	

- Create a function that takes a pointer to int as a parameter, and sets the value "42" to that int.
- Here's how it should be prototyped :

```
void        ft_ft(int *nbr);
```

Chapter IV

Exercise 01 : ft_ultimate_ft

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

- Create a function that takes a pointer to pointer to pointer to pointer to pointer to pointer to pointer to pointer to pointer to int as a parameter and sets the value "42" to that int.
- Here's how it should be prototyped :

```
void        ft_ultimate_ft(int *********nbr);
```

Chapter V

Exercise 02 : ft_swap

	Exercise 02
ft_swap	
Turn-in directory : $ex02/$	
Files to turn in : `ft_swap.c`	
Allowed functions : `None`	

- Create a function that swaps the value of two integers whose addresses are entered as parameters.
- Here's how it should be prototyped :

```
void    ft_swap(int *a, int *b);
```

Chapter VI

Exercise 03 : ft_div_mod

	Exercise 03
ft_div_mod	
Turn-in directory : *ex*03/	
Files to turn in : `ft_div_mod.c`	
Allowed functions : `None`	

- Create a function `ft_div_mod` prototyped like this :

```
void    ft_div_mod(int a, int b, int *div, int *mod);
```

- This function divides parameters `a` by `b` and stores the result in the int pointed by `div`. It also stores the remainder of the division of `a` by `b` in the int pointed by `mod`.

Chapter VII

Exercise 04 : ft_ultimate_div_mod

	Exercise 04
ft_ultimate_div_mod	
Turn-in directory : *ex*04/	
Files to turn in : `ft_ultimate_div_mod.c`	
Allowed functions : `None`	

- Create a function `ft_ultimate_div_mod` with the following prototype :

```
void    ft_ultimate_div_mod(int *a, int *b);
```

- This function divides parameters `a` by `b`. The result of this division is stored in the int pointed by `a`. The remainder of the division is stored in the int pointed by `b`.

Chapter VIII

Exercise 05 : ft_putstr

	Exercise 05
ft_putstr	
Turn-in directory : $ex05/$	
Files to turn in : `ft_putstr.c`	
Allowed functions : `write`	

- Create a function that displays a string of characters on the standard output.
- Here's how it should be prototyped :

```
void    ft_putstr(char *str);
```

Chapter IX

Exercise 06 : ft_strlen

	Exercise 06
ft_strlen	
Turn-in directory : $ex06/$	
Files to turn in : `ft_strlen.c`	
Allowed functions : `None`	

- Create a function that counts and returns the number of characters in a string.
- Here's how it should be prototyped :

```
int        ft_strlen(char *str);
```

Chapter X

Exercise 07 : ft_rev_int_tab

	Exercise 07
ft_rev_int_tab	
Turn-in directory : *ex*07/	
Files to turn in : `ft_rev_int_tab.c`	
Allowed functions : `None`	

- Create a function which reverses a given array of integer (first goes last, etc).
- The arguments are a pointer to int and the number of ints in the array.
- Here's how it should be prototyped :

```
void    ft_rev_int_tab(int *tab, int size);
```

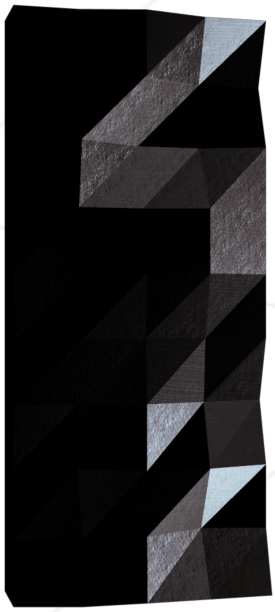
Chapter XI

Exercise 08 : ft_sort_int_tab

	Exercise 08
ft_sort_int_tab	
Turn-in directory : *ex*08/	
Files to turn in : `ft_sort_int_tab.c`	
Allowed functions : `None`	

- Create a function which sorts an array of integers by ascending order.
- The arguments are a pointer to int and the number of ints in the array.
- Here's how it should be prototyped :

```
void    ft_sort_int_tab(int *tab, int size);
```



C Piscine

C 02

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

Contents

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 `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

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

	Exercise 00
ft_strcpy	
Turn-in directory : $ex00/$	
Files to turn in : `ft_strcpy.c`	
Allowed functions : `None`	

- 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

	Exercise 01
ft_strncpy	
Turn-in directory : $ex01/$	
Files to turn in : `ft_strncpy.c`	
Allowed functions : `None`	

- 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

	Exercise 02
ft_str_is_alpha	
Turn-in directory : *ex*02/	
Files to turn in : `ft_str_is_alpha.c`	
Allowed functions : `None`	

- 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);
```

- It should return 1 if `str` is empty.

Chapter VI

Exercise 03 : ft_str_is_numeric

	Exercise 03
ft_str_is_numeric	
Turn-in directory : *ex*03/	
Files to turn in : `ft_str_is_numeric.c`	
Allowed functions : `None`	

- 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);
```

- It should return 1 if `str` is empty.

Chapter VII

Exercise 04 : ft_str_is_lowercase

	Exercise 04
ft_str_is_lowercase	
Turn-in directory : *ex*04/	
Files to turn in : `ft_str_is_lowercase.c`	
Allowed functions : `None`	

- 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);
```

- It should return 1 if `str` is empty.

Chapter VIII

Exercise 05 : ft_str_is_uppercase

	Exercise 05
ft_str_is_uppercase	
Turn-in directory : $ex05/$	
Files to turn in : `ft_str_is_uppercase.c`	
Allowed functions : `None`	

- 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);
```

- It should return 1 if `str` is empty.

Chapter IX

Exercise 06 : ft_str_is_printable

	Exercise 06
ft_str_is_printable	
Turn-in directory : *ex*06/	
Files to turn in : `ft_str_is_printable.c`	
Allowed functions : `None`	

- 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);
```

- It should return 1 if `str` is empty.

Chapter X

Exercise 07 : ft_strupcase

	Exercise 07
	ft_strupcase
Turn-in directory : $ex07/$	
Files to turn in : `ft_strupcase.c`	
Allowed functions : `None`	

- 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

	Exercise 08
ft_strlowcase	
Turn-in directory : *ex*08/	
Files to turn in : `ft_strlowcase.c`	
Allowed functions : `None`	

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

```
char        *ft_strlowcase(char *str);
```

- It should return `str`.

Chapter XII

Exercise 09 : ft_strcapitalize

	Exercise 09
ft_strcapitalize	
Turn-in directory : *ex*09/	
Files to turn in : `ft_strcapitalize.c`	
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_strlcpy.c`	
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

	Exercise 11
ft_putstr_with_non_printable	
Turn-in directory : $ex11/$	
Files to turn in : `ft_putstr_non_printable.c`	
Allowed functions : `write`	

- 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), preceeded by a "backslash".
- For example :

```
Coucou\ntu vas bien ?
```

- The function should display :

```
Coucou\0atu 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_print_memory.c`	
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` equals to 0, nothing should be displayed.

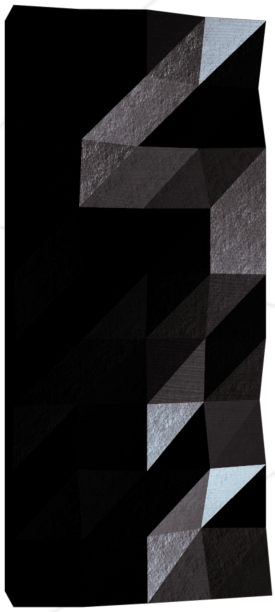
- Example:

```
$> ./ft_print_memory
000000010a161f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin
000000010a161f50: 6368 6573 090a 0963 2020 6573 7420 666f ches...c est fo
000000010a161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
000000010a161f70: 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. .
$> ./ft_print_memory | cat -te
0000000107ff9f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin$
0000000107ff9f50: 6368 6573 090a 0963 2020 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`.



C Piscine

C 03

Summary: This document is the subject for the C 03 module of the C Piscine @ 42.

Contents

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

Chapter II

Foreword

The first known mention of the game of RPS was in the book Wuzazu written by the Chinese Ming-dynasty writer Xie Zhaozhi who wrote that the game dated back to the time of the Chinese Han dynasty (206 BC – 220 AD). In the book, the game was called shoushiling. Li Rihua's book Note of Liuyanzhai also mentions this game, calling it shoushiling, huozhitou, or huoquan.

Throughout Japanese history there are frequent references to "sansukumi-ken", meaning "ken" fist games with a "san" three-way "sukumi" deadlock. This is in the sense that A beats B, B beats C, and C beats A. The games originated in China before being imported to Japan and subsequently becoming popular.

By the early 20th century, rock–paper–scissors had spread beyond Asia, especially through increased Japanese contact with the west. Its English-language name is therefore taken from a translation of the names of the three Japanese hand-gestures for rock, paper and scissors: elsewhere in Asia the open-palm gesture represents "cloth" rather than "paper". The shape of the scissors is also adopted from the Japanese style.

In 1927 La Vie au patronage, a children's magazine in France, described it in detail, referring to it as a "jeu japonais" ("Japanese game"). Its French name, "Chi-fou-mi", is based on the Old Japanese words for "one, two, three" ("hi, fu, mi")

A New York Times article of 1932 on the Tokyo rush hour describes the rules of the game for the benefit of American readers, suggesting it was not at that time widely known in the U.S. The 1933 edition of the Compton's Pictured Encyclopedia described it as a common method of settling disputes between children in its article on Japan; the name was given as "John Kem Po" and the article pointedly asserted, "This is such a good way of deciding an argument that American boys and girls might like to practice it too."

Chapter III

Exercise 00 : ft_strcmp

	Exercise 00
ft_strcmp	
Turn-in directory : $ex00/$	
Files to turn in : `ft_strcmp.c`	
Allowed functions : `None`	

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

```
int        ft_strcmp(char *s1, char *s2);
```

Chapter IV

Exercise 01 : ft_strncmp

	Exercise 01
ft_strncmp	
Turn-in directory : $ex01/$	
Files to turn in : `ft_strncmp.c`	
Allowed functions : `None`	

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

```
int        ft_strncmp(char *s1, char *s2, unsigned int n);
```

Chapter V

Exercise 02 : ft_strcat

	Exercise 02
ft_strcat	
Turn-in directory : *ex*02/	
Files to turn in : `ft_strcat.c`	
Allowed functions : `None`	

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

```
char *ft_strcat(char *dest, char *src);
```

Chapter VI

Exercise 03 : ft_strncat

	Exercise 03
ft_strncat	
Turn-in directory : $ex03/$	
Files to turn in : `ft_strncat.c`	
Allowed functions : `None`	

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

```
char *ft_strncat(char *dest, char *src, unsigned int nb);
```

Chapter VII

Exercise 04 : ft_strstr

	Exercise 04
ft_strstr	
Turn-in directory : *ex*04/	
Files to turn in : `ft_strstr.c`	
Allowed functions : `None`	

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

```
char        *ft_strstr(char *str, char *to_find);
```

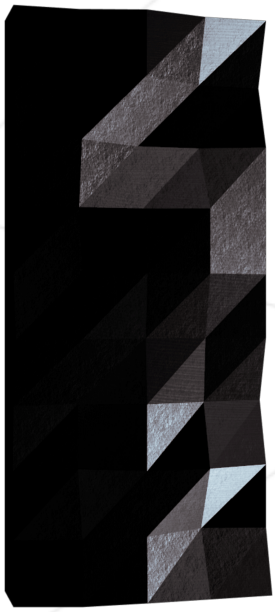
Chapter VIII

Exercise 05 : ft_strlcat

	Exercise 05
ft_strlcat	
Turn-in directory : $ex05/$	
Files to turn in : `ft_strlcat.c`	
Allowed functions : `None`	

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

```
unsigned int ft_strlcat(char *dest, char *src, unsigned int size);
```



C Piscine

C 04

Summary: this document is the subject for the unit C 04 of the C Piscine @ 42.

Contents

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

Here are the lyrics for `City Hunter`'s theme song "Moonlight Shadow":

```
The last time ever she saw him
Carried away by a moonlight shadow
He passed on worried and warning
Carried away by a moonlight shadow.
Lost in a riddle that Saturday night
Far away on the other side.
He was caught in the middle of a desperate fight
And she couldn't find how to push through

The trees that whisper in the evening
Carried away by a moonlight shadow
Sing a song of sorrow and grieving
Carried away by a moonlight shadow
All she saw was a silhouette of a gun
Far away on the other side.
He was shot six times by a man on the run
And she couldn't find how to push through

[Chorus]
I stay, I pray
See you in Heaven far away...
I stay, I pray
See you in Heaven one day.

Four A.M. in the morning
Carried away by a moonlight shadow
I watched your vision forming
Carried away by a moonlight shadow
A star was glowing in the silvery night
Far away on the other side
Will you come to talk to me this night
But she couldn't find how to push through

[Chorus]
```

```
Far away on the other side.
Caught in the middle of a hundred and five
The night was heavy and the air was alive
But she couldn't find how to push through
Carried away by a moonlight shadow
Carried away by a moonlight shadow
Far away on the other side.
```

Unfortunately, this topic has nothing to do with `City Hunter`.

Chapter III

Exercise 00 : ft_strlen

	Exercise 00
ft_strlen	
Turn-in directory : $ex00/$	
Files to turn in : `ft_strlen.c`	
Allowed functions : `None`	

- Create a function that counts and returns the number of characters in a string.
- Here's how it should be prototyped :

```
int    ft_strlen(char *str);
```

Chapter IV

Exercise 01 : ft_putstr

	Exercise 01
ft_putstr	
Turn-in directory : $ex01/$	
Files to turn in : `ft_putstr.c`	
Allowed functions : `write`	

- Create a function that displays a string of characters on the standard output.
- Here's how it should be prototyped :

```
void    ft_putstr(char *str);
```

Chapter V

Exercise 02 : ft_putnbr

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

- 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 VI

Exercise 03 : ft_atoi

	Exercise 03
ft_atoi	
Turn-in directory : *ex*03/	
Files to turn in : `ft_atoi.c`	
Allowed functions : `None`	

- Write a function that converts the initial portion of the string pointed by str to its int representation
- The string can start with an arbitray amount of white space (as determined by `isspace(3)`)
- The string can be followed by an arbitrary amount of + and - signs, - sign will change the sign of the int returned based on the number of - is odd or even.
- Finally the string can be followed by any numbers of the base 10.
- Your function should read the string until the string stop following the rules and return the number found until now.
- You should not take care of overflow or underflow. result can be undefined in that case.
- Here's an example of a program that prints the atoi return value:

```
$>./a.out "   ---+--+1234ab567"
-1234
```

- Here's how it should be prototyped :

```
int     ft_atoi(char *str);
```

Chapter VII

Exercise 04 : ft_putnbr_base

	Exercise 04
ft_putnbr_base	
Turn-in directory : *ex*04/	
Files to turn in : `ft_putnbr_base.c`	
Allowed functions : `write`	

- Create a function that displays a number in a base system in the terminal.
- This number is given in the shape of an `int`, and the radix in the shape of a `string of characters`.
- The base-system contains all useable symbols to display that number :
  - `0123456789` is the commonly used base system to represent decimal numbers
  - `01` is a binary base system ;
  - `0123456789ABCDEF` an hexadecimal base system ;
  - `poneyvif` is an octal base system.
- The function must handle negative numbers.
- If there's an invalid argument, nothing should be displayed. Examples of invalid arguments :
  - base is empty or size of 1;
  - base contains the same character twice ;
  - base contains + or - ;
- Here's how it should be prototyped :

```
void        ft_putnbr_base(int nbr, char *base);
```

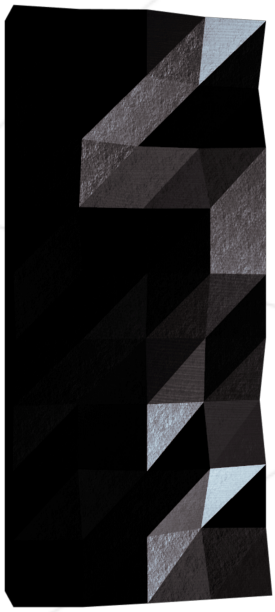
Chapter VIII

Exercise 05 : ft_atoi_base

	Exercise 05
ft_atoi_base	
Turn-in directory : $ex05/$	
Files to turn in : `ft_atoi_base.c`	
Allowed functions : `None`	

- Write a function that converts the initial portion of the string pointed by str to int representation.
- str is in a specific base given as a second parameter.
- excepted the base rule, the function should work exactly like ft_atoi.
- If there's an invalid argument, the function should return 0. Examples of invalid arguments :
  - base is empty or size of 1;
  - base contains the same character twice ;
  - base contains + or - or whitespaces;
- Here's how it should be prototyped :

```
int        ft_atoi_base(char *str, char *base);
```



C Piscine

C 05

Summary: this document is the subject for the C 05 module of the C Piscine @ 42.

Contents

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

Here are some lyrics extract from the `Harry Potter` saga:

```
Oh you may not think me pretty,
But don't judge on what you see,
I'll eat myself if you can find
A smarter hat than me.

You can keep your bowlers black,
Your top hats sleek and tall,
For I'm the Hogwarts Sorting Hat
And I can cap them all.

The Sorting Hat, stored in the Headmaster's Office.
There's nothing hidden in your head
The Sorting Hat can't see,
So try me on and I will tell you
Where you ought to be.

You might belong in Gryffindor,
Where dwell the brave at heart,
Their daring, nerve, and chivalry
Set Gryffindors apart;

You might belong in Hufflepuff,
Where they are just and loyal,
Those patient Hufflepuffs are true
And unafraid of toil;

Or yet in wise old Ravenclaw,
If you've a ready mind,
Where those of wit and learning,
Will always find their kind;

Or perhaps in Slytherin
You'll make your real friends,
Those cunning folks use any means
```

```
To achieve their ends.

So put me on! Don't be afraid!
And don't get in a flap!
You're in safe hands (though I have none)
For I'm a Thinking Cap!
```

Unfortunately, this subject's got nothing to do with the `Harry Potter` saga, which is too bad, because your exercises won't be done by `magic`.

Chapter III

Exercise 00 : ft_iterative_factorial

	Exercise 00
ft_iterative_factorial	
Turn-in directory : *ex*00/	
Files to turn in : `ft_iterative_factorial.c`	
Allowed functions : `None`	

- Create an iterated function that returns a number. This number is the result of a factorial operation based on the number given as a parameter.
- If the argument is not valid the function should return 0.
- Overflows must not be handled, the function return will be undefined.
- Here's how it should be prototyped :

```
int ft_iterative_factorial(int nb);
```

Chapter IV

Exercise 01 : ft_recursive_factorial

	Exercise 01
ft_recursive_factorial	
Turn-in directory : $ex01/$	
Files to turn in : `ft_recursive_factorial.c`	
Allowed functions : `None`	

- Create a recursive function that returns the factorial of the number given as a parameter.
- If the argument is not valid the function should return 0.
- Overflows must not be handled, the function return will be undefined.
- Here's how it should be prototyped :

```
int ft_recursive_factorial(int nb);
```

Chapter V

Exercise 02 : ft_iterative_power

	Exercise 02
ft_iterative_power	
Turn-in directory : $ex02/$	
Files to turn in : `ft_iterative_power.c`	
Allowed functions : `None`	

- Create an iterated function that returns the value of a power applied to a number. An power lower than 0 returns 0. Overflows must not be handled.
- We've decided that 0 power 0 will returns 1
- Here's how it should be prototyped :

```
int ft_iterative_power(int nb, int power);
```

Chapter VI

Exercise 03 : ft_recursive_power

	Exercise 03
ft_recursive_power	
Turn-in directory : *ex*03/	
Files to turn in : `ft_recursive_power.c`	
Allowed functions : `None`	

- Create a recursive function that returns the value of a power applied to a number.
- Overflows must not be handled, the function return will be undefined.
- We've decided that 0 power 0 will returns 1
- Here's how it should be prototyped :

```
int ft_recursive_power(int nb, int power);
```

Chapter VII

Exercise 04 : ft_fibonacci

	Exercise 04
ft_fibonacci	
Turn-in directory : *ex*04/	
Files to turn in : `ft_fibonacci.c`	
Allowed functions : `None`	

- Create a function `ft_fibonacci` that returns the `n`-th element of the Fibonacci sequence, the first element being at the 0 index. We'll consider that the Fibonacci sequence starts like this: 0, 1, 1, 2.
- Overflows must not be handled, the function return will be undefined.
- Here's how it should be prototyped :

```
int ft_fibonacci(int index);
```

- Obviously, `ft_fibonacci` has to be recursive.
- If the `index` is less than 0, the function should return -1.

Chapter VIII

Exercise 05 : ft_sqrt

	Exercise 05
ft_sqrt	
Turn-in directory : $ex05/$	
Files to turn in : `ft_sqrt.c`	
Allowed functions : `None`	

- Create a function that returns the square root of a number (if it exists), or 0 if the square root is an irrational number.
- Here's how it should be prototyped :

```
int ft_sqrt(int nb);
```

Chapter IX

Exercise 06 : ft_is_prime

	Exercise 06
ft_is_prime	
Turn-in directory : *ex*06/	
Files to turn in : `ft_is_prime.c`	
Allowed functions : `None`	

- Create a function that returns 1 if the number given as a parameter is a prime number, and 0 if it isn't.
- Here's how it should be prototyped :

```
int ft_is_prime(int nb);
```

`0 and 1 are not prime numbers.`

Chapter X

Exercise 07 : ft_find_next_prime

	Exercise 07
ft_find_next_prime	
Turn-in directory : $ex07/$	
Files to turn in : `ft_find_next_prime.c`	
Allowed functions : `None`	

- Create a function that returns the next prime number greater or equal to the number given as argument.
- Here's how it should be prototyped :

```
int ft_find_next_prime(int nb);
```

Chapter XI

Exercise 08 : The Ten Queens

	Exercise 08
The Ten Queens	
Turn-in directory : $ex08/$	
Files to turn in : `ft_ten_queens_puzzle.c`	
Allowed functions : `write`	

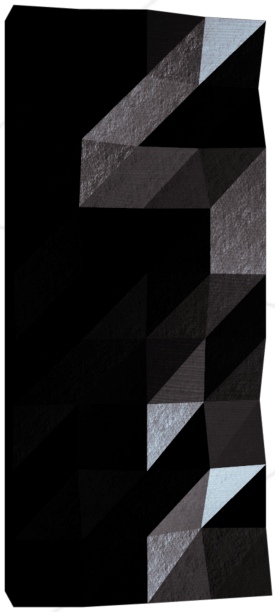
- Create a function that displays all possible placements of the ten queens on a chessboard which would contain ten columns and ten lines, without them being able to reach each other in a single move, and returns the number of possibilities.
- Recursivity is required to solve this problem.
- Here's how it should be prototyped :

```
int ft_ten_queens_puzzle(void);
```

- Here's how it'll be displayed :

```
$>./a.out | cat -e
0257948136$
0258693147$
...
4605713829$
4609582731$
...
9742051863$
$>
```

- The sequence goes from left to right. The first digit represents the first Queen's position in the first column (the index starting from 0). The Nth digit represents the Nth Queen's position in the Nth column.
- The return value must be the total number of displayed solutions.



C Piscine

C 06

Staff 42 pedago@42.fr

Summary: This document is the subject for the module C 06 of the C Piscine @ 42.

Contents

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 : ft_print_program_name

	Exercise 00
ft_print_program_name	
Turn-in directory : $ex00/$	
Files to turn in : `ft_print_program_name.c`	
Allowed functions : `write`	

- 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 IV

Exercise 01 : ft_print_params

	Exercise 01
ft_print_params	
Turn-in directory : $ex01/$	
Files to turn in : `ft_print_params.c`	
Allowed functions : `write`	

- 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.
- One per line, in the same order as in the command line.
- It should display all arguments, except for `argv[0]`.
- Example :

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

Chapter V

Exercise 02 : ft_rev_params

	Exercise 02
ft_rev_params	
Turn-in directory : $ex02/$	
Files to turn in : `ft_rev_params.c`	
Allowed functions : `write`	

- 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.
- One per line, in the reverse order of the command line.
- It should display all arguments, except for `argv[0]`.

Chapter VI

Exercise 03 : ft_sort_params

	Exercise 03
ft_sort_params	
Turn-in directory : $ex03/$	
Files to turn in : `ft_sort_params.c`	
Allowed functions : `write`	

- 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]`.
- One argument per line.