Piscine

03

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

Contents

Ch pter 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? sk your peer on the right. Otherwise, try your peer on the left.

Your reference guide is called `Google / m n / 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 l unched with the -R CheckForbiddenSourceHe der fl g. Moulinette will use it too.`

Ch pter 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 D). 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 beats B, B beats C, and C beats . 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 sia, 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 sia 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")

New York Times article of 1932 on the Tokyo rush hour describes the rules of the game for the benefit of merican 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 merican boys and girls might like to practice it too."

Ch pter III

Exercise 00 : ft_strcmp

	Exercise 00
ft_strcmp	
Turn-in directory : *ex*00	
Files to turn in : `ft_strcmp.c`	
llowed functions : `None`	

Reproduce the behavior of the function `strcmp` (man strcmp).

Here's how it should be prototyped :

```
int        ft_strcmp(ch r *s1, ch r *s2);
```

Ch pter IV

Exercise 01 : ft_strncmp

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

Reproduce the behavior of the function `strncmp` (man strncmp).

Here's how it should be prototyped :

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

Ch pter V

Exercise 02 : ft_strc t

	Exercise 02
ft_strcat	
Turn-in directory : *ex*02	
Files to turn in : `ft_strc t.c`	
llowed functions : `None`	

Reproduce the behavior of the function `strc t` (man strcat).

Here's how it should be prototyped :

```
ch r *ft_strc t(ch r *dest, ch r *src);
```

Ch pter VI

Exercise 03 : ft_strnc t

	Exercise 03
ft_strncat	
Turn-in directory : *ex*03	
Files to turn in : `ft_strnc t.c`	
llowed functions : `None`	

Reproduce the behavior of the function `strnc t` (man strncat).

Here's how it should be prototyped :

```
ch r *ft_strnc t(ch r *dest, ch r *src, unsigned int nb);
```

Ch pter VII

Exercise 04 : ft_strstr

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

Reproduce the behavior of the function `strstr` (man strstr).

Here's how it should be prototyped :

```
ch r        *ft_strstr(ch r *str, ch r *to_find);
```

Ch pter VIII

Exercise 05 : ft_strlc t

	Exercise 05
ft_strlcat	
Turn-in directory : $ex05$	
Files to turn in : `ft_strlc t.c`	
llowed functions : `None`	

Reproduce the behavior of the function `strlc t` (man strlcat).

Here's how it should be prototyped :

```
unsigned int ft_strlc t(ch r *dest, ch r *src, unsigned int size);
```