Piscine Reloaded

It’s good to be back

*Summary:*

*The Piscine was good but the time has past. This serie of exercises will help you to remind all the basics you’ve learned during the piscine. Functions, loops, pointers, structures, let’s remind together the syntactic and semantic bases of the C*

*Version: 1.2*

**Contents**

**I Foreword 2 II Introduction 3 III General rules 4 IV Exercise 00 : Oh yeah, mooore... 6 V Exercise 01 : Z 7 VI Exercise 02 : clean 8 VII Exercise 03 : find\_sh 9 VIII Exercise 04 : MAC 10 IX Exercise 05 : Can you create it ? 11 X Exercise 06 : ft\_print\_alphabet 12 XI Exercise 07 : ft\_print\_numbers 13 XII Exercise 08 : ft\_is\_negative 14 XIII Exercise 09 : ft\_ft 15 XIV Exercise 10 : ft\_swap 16 XV Exercise 11 : ft\_div\_mod 17 XVI Exercise 12 : ft\_iterative\_factorial 18 XVII Exercise 13 : ft\_recursive\_factorial 19 XVIII Exercise 14 : ft\_sqrt 20 XIX Exercise 15 : ft\_putstr 21 XX Exercise 16 : ft\_strlen 22 XXI Exercise 17 : ft\_strcmp 23 XXII Exercise 18 : ft\_print\_params 24 XXIII Exercise 19 : ft\_sort\_params 25**

1

Piscine Reloaded It’s good to be back

**XXIV Exercise 20 : ft\_strdup 26 XXV Exercise 21 : ft\_range 27 XXVI Exercise 22 : ft\_abs.h 28 XXVII Exercise 23 : ft\_point.h 29 XXVIIIExercise 24 : Makefile 30 XXIX Exercise 25 : ft\_foreach 31 XXX Exercise 26 : ft\_count\_if 32 XXXI Exercise 27 : display\_file 33 XXXII Submission and peer-evaluation 34**

2

**Chapter I**

**Foreword**

Edward Joseph Snowden (born June 21, 1983) is an American computer professional, for mer Central Intelligence Agency (CIA) employee, and former contractor for the United States government who copied and leaked classified information from the National Se curity Agency (NSA) in 2013 without authorization. His disclosures revealed numerous global surveillance programs, many run by the NSA and the Five Eyes Intelligence Al liance with the cooperation of telecommunication companies and European governments.

In 2013, Snowden was hired by an NSA contractor, Booz Allen Hamilton, after previ ous employment with Dell and the CIA. On May 20, 2013, Snowden flew to Hong Kong after leaving his job at an NSA facility in Hawaii, and in early June he revealed thousands of classified NSA documents to journalists Glenn Greenwald, Laura Poitras, and Ewen MacAskill. Snowden came to international attention after stories based on the material appeared in The Guardian and The Washington Post. Further disclosures were made by other publications including Der Spiegel and The New York Times.

On June 21, 2013, the U.S. Department of Justice unsealed charges against Snowden of two counts of violating the Espionage Act of 1917 and theft of government property. Two days later, he flew into Moscow’s Sheremetyevo Airport, but Russian authorities noted that his U.S. passport had been cancelled and he was restricted to the airport terminal for over one month. Russia ultimately granted him right of asylum for one year, and repeated extensions have permitted him to stay at least until 2020. He reportedly lives in an undisclosed location in Moscow, and continues to seek asylum elsewhere in the world.

A subject of controversy, Snowden has been variously called a hero, a whistleblower, a dissident, a traitor and a patriot. His disclosures have fueled debates over mass surveil lance, government secrecy, and the balance between national security and information privacy.

If you’d like to find out more, we recommend you watch the documentary Citizenfour. 3

**Chapter II**

**Introduction**

The Piscine Reloaded is a best-of of the exercises you did during the C Piscine to remind you all the basics of the C programming language.

If you have already done some of these exercises during the Piscine C, we highly recommend not be tempted to retrieve your old code. The learning of programming involves practice and making an existing code has no interest.

4

**Chapter III**

**General rules**

*•* Only this page will serve as reference; do not trust rumors.

*•* Make sure you have the appropriate permissions on your files and directories. *•* You have to follow the turn-in procedures for every exercise.

*•* Your exercises will **only** 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.

*•* Exercises in Shell must be executable with /bin/sh.

*•* 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 / ....

*•* Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...

*•* 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.

*•* 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 ft\_putchar() is an authorized function, we will compile your code with our ft\_putchar.c.

*•* If your program doesn’t compile, you’ll get 0.

5

Piscine Reloaded It’s good to be back *•* By Odin, by Thor! Use your brain!!!

6

**Chapter IV**

**Exercise 00 : Oh yeah, mooore...**

|  | Exercise 00 |
| --- | --- |
| Oh yeah, mooore... | |
| Turn-in directory : *ex*00*/* | |
| Files to turn in : exo.tar | |
| Allowed functions : None | |

*•* Create the following files and directories. Do what’s necessary so that when you use the ls -l command in your directory, the output will looks like this :

%> ls -l

total XX

drwx--xr-x 2 XX XX XX Jun 1 20:47 test0

-rwx--xr-- 1 XX XX 4 Jun 1 21:46 test1

dr-x---r-- 2 XX XX XX Jun 1 22:45 test2

-r-----r-- 2 XX XX 1 Jun 1 23:44 test3

-rw-r----x 1 XX XX 2 Jun 1 23:43 test4

-r-----r-- 2 XX XX 1 Jun 1 23:44 test5

lrwxrwxrwx 1 XX XX 5 Jun 1 22:20 test6 -> test0

%>

*•* About the hours, it will be accepted if the year is diplayed in the case of the exercise’s date (1 Jun) is outdated by six month or more.

*•* Once you’ve done that, run tar -cf exo.tar \* to create the file to be submitted. Don’t worry about what you’ve got instead of "XX".

7

**Chapter V**

**Exercise 01 : Z**

|  | Exercise 01 |
| --- | --- |
| Only the best know how to display Z | |
| Turn-in directory : *ex*01*/* | |
| Files to turn in : z | |
| Allowed functions : None | |

*•* Create a file called z that returns "Z", followed by a new line, whenever the command cat is used on it.

?>cat z

Z

?>

8

**Chapter VI**

**Exercise 02 : clean**

|  | Exercise 02 |
| --- | --- |
|  | |
| Turn-in directory : *ex*02*/* | |
| Files to turn in : clean | |
| Allowed functions : None | |

*•* In a file called clean place the command line that will search for all files - in the current directory as well as in its sub-directories - with a name ending by ~, or with a name that start and end by #

*•* The command line will show and erase all files found.

*•* Only one command is allowed: no ’;’ or ’&&’ or other shenanigans. man find

9

**Chapter VII**

**Exercise 03 : find\_sh**

|  | Exercise 03 |
| --- | --- |
| find\_sh.sh | |
| Turn-in directory : *ex*03*/* | |
| Files to turn in : find\_sh.sh | |
| Allowed functions : None | |

*•* Write a command line that searches for all file names that end with ".sh" (without quotation marks) in the current directory and all its sub-directories. It should display only the file names without the .sh.

*•* Example of output :

$>./find\_sh.sh | cat -e

find\_sh$

file1$

file2$

file3$

$>

10

**Chapter VIII**

**Exercise 04 : MAC**

|  | Exercise 04 |
| --- | --- |
| MAC.sh | |
| Turn-in directory : *ex*04*/* | |
| Files to turn in : MAC.sh | |
| Allowed functions : None | |

*•* Write a command line that displays your machine’s MAC addresses. Each address must be followed by a line break.

man ifconfig

11

**Chapter IX**

**Exercise 05 : Can you create it ?**

|  | Exercise 05 |
| --- | --- |
| Can you create it ? | |
| Turn-in directory : *ex*05*/* | |
| Files to turn in : "\?$\*’MaRViN’\*$?\" | |
| Allowed functions : None | |

*•* Create a file containing only "42", and NOTHING else.

*•* Its name will be :

"\?$\*'MaRViN'\*$?\"

*•* Example :

$>ls -lRa \*MaRV\* | cat -e

-rw---xr-- 1 75355 32015 2 Oct 2 12:21 "\?$\*'MaRViN'\*$?\"$

$>

12

**Chapter X**

**Exercise 06 : ft\_print\_alphabet**

|  | Exercise 06 |
| --- | --- |
| ft\_print\_alphabet | |
| Turn-in directory : *ex*06*/* | |
| Files to turn in : ft\_print\_alphabet.c | |
| Allowed functions : ft\_putchar | |

*•* 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**);

13

**Chapter XI**

**Exercise 07 : ft\_print\_numbers**

|  | Exercise 07 |
| --- | --- |
| ft\_print\_numbers | |
| Turn-in directory : *ex*07*/* | |
| Files to turn in : ft\_print\_numbers.c | |
| Allowed functions : ft\_putchar | |

*•* 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**);

14

**Chapter XII**

**Exercise 08 : ft\_is\_negative**

|  | Exercise 08 |
| --- | --- |
| ft\_is\_negative | |
| Turn-in directory : *ex*08*/* | |
| Files to turn in : ft\_is\_negative.c | |
| Allowed functions : ft\_putchar | |

*•* 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);

15

**Chapter XIII**

**Exercise 09 : ft\_ft**

|  | Exercise 09 |
| --- | --- |
| ft\_ft | |
| Turn-in directory : *ex*09*/* | |
| 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);

16

**Chapter XIV**

**Exercise 10 : ft\_swap**

|  | Exercise 10 |
| --- | --- |
| ft\_swap | |
| Turn-in directory : *ex*10*/* | |
| 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);

17

**Chapter XV**

**Exercise 11 : ft\_div\_mod**

|  | Exercise 11 |
| --- | --- |
| ft\_div\_mod | |
| Turn-in directory : *ex*11*/* | |
| 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.

18

**Chapter XVI**

**Exercise 12 : ft\_iterative\_factorial**

|  | Exercise 12 |
| --- | --- |
| ft\_iterative\_factorial | |
| Turn-in directory : *ex*12*/* | |
| 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 there’s an error, the function should return 0.

*•* Here’s how it should be prototyped :

**int** ft\_iterative\_factorial(**int** nb);

*•* Your function must return its result in less than two seconds.

19

**Chapter XVII**

**Exercise 13 : ft\_recursive\_factorial**

|  | Exercise 13 |
| --- | --- |
| ft\_recursive\_factorial | |
| Turn-in directory : *ex*13*/* | |
| 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 there’s an error, the function should return 0.

*•* Here’s how it should be prototyped :

**int** ft\_recursive\_factorial(**int** nb);

20

**Chapter XVIII**

**Exercise 14 : ft\_sqrt**

|  | Exercise 14 |
| --- | --- |
| ft\_sqrt | |
| Turn-in directory : *ex*14*/* | |
| 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);

*•* Your function must return its result in less than two seconds.

21

**Chapter XIX**

**Exercise 15 : ft\_putstr**

|  | Exercise 15 |
| --- | --- |
| ft\_putstr | |
| Turn-in directory : *ex*15*/* | |
| Files to turn in : ft\_putstr.c | |
| Allowed functions : ft\_putchar | |

*•* 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);

22

**Chapter XX**

**Exercise 16 : ft\_strlen**

|  | Exercise 16 |
| --- | --- |
| ft\_strlen | |
| Turn-in directory : *ex*16*/* | |
| Files to turn in : ft\_strlen.c | |
| Allowed functions : None | |

*•* Reproduce the behavior of the function strlen (man strlen).

*•* Here’s how it should be prototyped :

**int** ft\_strlen(**char** \*str);

23

**Chapter XXI**

**Exercise 17 : ft\_strcmp**

|  | Exercise 17 |
| --- | --- |
| ft\_strcmp | |
| Turn-in directory : *ex*17*/* | |
| 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);

24

**Chapter XXII**

**Exercise 18 : ft\_print\_params**

|  | Exercise 18 |
| --- | --- |
| ft\_print\_params | |
| Turn-in directory : *ex*18*/* | |
| Files to turn in : ft\_print\_params.c | |
| Allowed functions : ft\_putchar | |

*•* 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

$>

25

**Chapter XXIII**

**Exercise 19 : ft\_sort\_params**

|  | Exercise 19 |
| --- | --- |
| ft\_sort\_params | |
| Turn-in directory : *ex*19*/* | |
| Files to turn in : ft\_sort\_params.c | |
| Allowed functions : ft\_putchar | |

*•* 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.

26

**Chapter XXIV**

**Exercise 20 : ft\_strdup**

|  | Exercise 20 |
| --- | --- |
| ft\_strdup | |
| Turn-in directory : *ex*20*/* | |
| Files to turn in : ft\_strdup.c | |
| Allowed functions : malloc | |

*•* Reproduce the behavior of the function strdup (man strdup).

*•* Here’s how it should be prototyped :

**char** \*ft\_strdup(**char** \*src);

27

**Chapter XXV**

**Exercise 21 : ft\_range**

|  | Exercise 21 |
| --- | --- |
| ft\_range | |
| Turn-in directory : *ex*21*/* | |
| Files to turn in : ft\_range.c | |
| Allowed functions : malloc | |

*•* Create a function ft\_range which returns an array ofints. This int array should contain all values between min and max.

*•* Min included - max excluded.

*•* Here’s how it should be prototyped :

**int** \*ft\_range(**int** min, **int** max);

*•* If min´value is greater or equal to max’s value, a null pointer should be returned. 28

**Chapter XXVI**

**Exercise 22 : ft\_abs.h**

|  | Exercise 22 |
| --- | --- |
| ft\_abs.h | |
| Turn-in directory : *ex*22*/* | |
| Files to turn in : ft\_abs.h | |
| Allowed functions : None | |

*•* Create a macro ABS which replaces its argument by it absolute value : *#define ABS(Value)*

You are asked to do something that is normally banned by the Norm, that will be the only time we autorize it.

29

**Chapter XXVII**

**Exercise 23 : ft\_point.h**

|  | Exercise 23 |
| --- | --- |
| ft\_point.h | |
| Turn-in directory : *ex*23*/* | |
| Files to turn in : ft\_point.h | |
| Allowed functions : None | |

*•* Create a file ft\_point.h that’ll compile the following main :

*#include "ft\_point.h"*

**void** set\_point(t\_point \*point)

{

point->x = 42;

point->y = 21;

}

**int** main(**void**)

{

t\_point point;

set\_point(&point);

**return** (0);

}

30

**Chapter XXVIII**

**Exercise 24 : Makefile**

|  | Exercise 24 |
| --- | --- |
| Makefile | |
| Turn-in directory : *ex*24*/* | |
| Files to turn in : Makefile | |
| Allowed functions : None | |

*•* Create the Makefile that’ll compile your libft.a.

*•* The Makefile will get its source files from the "srcs" directory.

*•* The Makefile will get its header files from the "includes" directory. *•* The lib will be at the root of the exercise.

*•* The Makefile should also implement the following rules: clean, fclean and re as well as all.

*•* fclean does the equivalent of a make clean and also erases the binary created during the make. re does the equivalent of a make fclean followed by a make.

*•* We’ll only fetch your Makefile and test it with our files. For this exercise, only the following 5 mandatory functions of your lib have to be handled : (ft\_putchar, ft\_putstr, ft\_strcmp, ft\_strlen and ft\_swap).

Watch out for wildcards!

31

**Chapter XXIX**

**Exercise 25 : ft\_foreach**

|  | Exercise 25 |
| --- | --- |
| ft\_foreach | |
| Turn-in directory : *ex*25*/* | |
| Files to turn in : ft\_foreach.c | |
| Allowed functions : None | |

*•* Create the function ft\_foreach which, for a given ints array, applies a function on all elements of the array. This function will be applied following the array’s order.

*•* Here’s how the function should be prototyped :

**void** ft\_foreach(**int** \*tab, **int** length, **void** (\*f)(**int**));

*•* For example, the function ft\_foreach could be called as follows in order to display all ints of the array :

ft\_foreach(tab, 1337, &ft\_putnbr);

32

**Chapter XXX**

**Exercise 26 : ft\_count\_if**

|  | Exercise 26 |
| --- | --- |
| ft\_count\_if | |
| Turn-in directory : *ex*26*/* | |
| Files to turn in : ft\_count\_if.c | |
| Allowed functions : None | |

*•* Create a function ft\_count\_if which will return the number of elements of the array that return 1, passed to the function f.

*•* Here’s how the function should be prototyped :

**int** ft\_count\_if(**char** \*\*tab, **int** (\*f)(**char**\*));

*•* The array will be delimited by 0.

33

**Chapter XXXI**

**Exercise 27 : display\_file**

|  | Exercise 27 |
| --- | --- |
| display\_file | |
| Turn-in directory : *ex*27*/* | |
| Files to turn in : Makefile, and files needed for your program | |
| Allowed functions : close, open, read, write | |

*•* Create a program called ft\_display\_file that displays, on the standard output, only the content of the file given as argument.

*•* The submission directory should have a Makefile with the following rules : all, clean, fclean. The binary will be called ft\_display\_file.

*•* The malloc function is forbidden. You can only do this exercise by declaring a fixed-sized array.

*•* All files given as arguments will be valid.

*•* Error messages have to be displayed on their reserved output followed by a new line.

*•* If no argument is given, it should display

File name missing.

*•* If there is more than one argument, it should display

Too many arguments.

*•* If the file cannot be read, it should display

Cannot read file.

34

**Chapter XXXII**

**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 folders and files to ensure they are correct.

35