Chapter III

ft_generic.c

	Exercise 00
ft_generic.c	
Turn-in directory : $ex00/$	
Files to turn in : `ft_generic.c`	
Allowed functions : `write`	
Notes : `n/a`	

- Create a function `ft_generic()`. When called by a main(), it should display :

```
$>./a.out
Tut tut ; Tut tut
$>
```

- Here's how it should be prototyped :

```
void  ft_generic(void);
```

Chapter III

ft_takes_place.c

`Even in the future, watches provide the time.`
`You just have to know how to set them up.`

	Exercise 01
ft_takes_place.c	
Turn-in directory : *ex*01/	
Files to turn in : `ft_takes_place.c`	
Allowed functions : `write, printf`	
Notes : `n/a`	

- Create a function `ft_takes_place` which displays the time on the standard output like this : `THE FOLLOWING TAKES PLACE BETWEEN X.00 A.M. AND Y.00 A.M.`. Followed by a line break. Example :

```
THE FOLLOWING TAKES PLACE BETWEEN 10.00 A.M. AND 11.00 A.M.
```

- Here's how it should be prototyped :

```
void  ft_takes_place(int hour);
```

- The `hour` argument will be a valid `int`, in a 24-hour clock format. The output will be in the am/pm format.

`Be careful with noon and midnight !`

Chapter III

find_nicolas_bauer.sh

	Exercise 02
find_nicolas_bauer.sh	
Turn-in directory : *ex*02/	
Files to turn in : `find_nicolas_bauer.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a file `find_nicolas_bauer.sh` which will display the number for all Nicolas Bauer from the phonebook passed as argument.
- Your program will be called with `sh find_nicolas_bauer.sh phonebook`

`You'll probably miss a few !`

Chapter III

defuse.sh

	Exercise 03
defuse.sh	
Turn-in directory : $ex03/$	
Files to turn in : `defuse.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a file `defuse.sh` which will substract a second off the `timestamp` (time since Epoch in seconds) from the last access to the file `bomb.txt`, and display it on the stadard output, followed by a line break. The `bomb.txt` file will be supplied in the same folder.

- No imposed limits of shell command-line usage for this exercise. Go nuts, be creative!

Chapter III

ft_rot42.c

	Exercise 04
ft_rot42.c	
Turn-in directory : *ex*04/	
Files to turn in : `ft_rot42.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_rot42` which takes the string passed as argument and rotates the appropriate characters by 42.
- Here's how it should be prototyped :

```
char  *ft_rot42(char *str);
```

If you're not sure what you're supposed to do here, know that rot13 already exists...

Chapter III

ft_antidote.c

	Exercise 05
ft_antidote.c	
Turn-in directory : $ex05/$	
Files to turn in : `ft_antidote.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_antidote` which takes three `ints` as argument and returns the middle value.
- Here's how it should be prototyped :

```
int ft_antidote(int i, int j, int k);
```

To know more about the middle value, read the foreword section...

Chapter III

ft_destroy.c

	Exercise 06
ft_destroy.c	
Turn-in directory : *ex*06/	
Files to turn in : `ft_destroy.c`	
Allowed functions : `free`	
Notes : `n/a`	

- Create a function `ft_destroy()` which will destroy (By calling the function `free`) the factory passed as argument.
- Here's how it should be prototyped :

```
void        ft_destroy(char ***factory);
```

- You'll have to include the file header `ft_ultimator.h`.
- The factory is smartly delimited by `NULL`s.

```
%>cat ft_ultimator.h
#ifndef __FT_ULTIMATOR_H__
# define __FT_ULTIMATOR_H__

/*
**
** With Windows VISTA, we were on the edge of the abyss.
** With Windows 8, we made a huge step forward.
**
** The Client: 'I have a computer running on Windows 8'
** The Technician: 'Yes...'
** The Client: 'And it doesn't work anymore'
** The Technician: 'Yeah, you already said...'
**
*/

#endif
%>
```

Chapter III

ft_collatz_conjecture.c

	Exercise 07
ft_collatz_conjecture.c	
Turn-in directory : *ex*07/	
Files to turn in : `ft_collatz_conjecture.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_collatz_conjecture` which will return the "flight time" for a given argument.
- This function must be recursive.
- Here's how it should be prototyped :

```
unsigned int  ft_collatz_conjecture(unsigned int base);
```

Chapter III

ft_spy.c

	Exercise 08
ft_spy.c	
Turn-in directory : *ex*08/	
Files to turn in : `ft_spy.c`	
Allowed functions : `write`	
Notes : `n/a`	

- Create the file `ft_spy.c` which displays `Alert!!!` on the standard output, followed by a line break, if any of the given arguments is one of these words: `president`, `attack` or `Bauer`.

- Evidently, your ft_spy.c file should have a `main`.

```
$> $./ft_spy "I" "will" "kill" "Nicolas" "Bauer"
  Alert!!!
  $> $./ft_spy "Hello" "I" "want" "a" "4" "cheese" "pizza"
  $> $./ft_spy "I" "will kill" "the president" "soon"
$> $./ft_spy "I" "will" "kill" "the" "president" "soon"
  Alert!!!
  $>
```

- Examine the examples thoroughly :

```
$> $./ft_spy "I" "will" "kill" "Jack" "BaUer"
  Alert!!!
  $> $./ft_spy "   the   " "  president    " "must" "die   "
  Alert!!!
  $> $./ft_spy "   the'" "omnipresident" "must" "eat   "
  $>
```

Why not take advantage of this exercise to add reuseable functions to your lib ?

Chapter III

where_am_i.sh

	Exercise 09
where_am_i.sh	
Turn-in directory : *ex*09/	
Files to turn in : `where_am_i.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a file `where_am_i.sh` which displays the IP(s) of the computer on the network (in IPV4 format), on the standard output. One IP per line.
- If no IP address is found, display `I am lost!` followed by a line break.

Chapter III

ft_scrambler.c

	Exercise 10
ft_scrambler.c	
Turn-in directory : *ex*10/	
Files to turn in : `ft_scrambler.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_scrambler()` that exchanges the `int`s pointed by pointers to `int` given as arguments.
- This function will put `a` in `c` ; `c` in `d` ; `d` in `b` ; and `b` in `a`.
- Here's how it should be prototyped :

```
void  ft_scrambler(int ***a, int *b, int *******c, int ****d);
```

Chapter III

ft_perso.h

	Exercise 11
ft_perso.h	
Turn-in directory : $ex11/$	
Files to turn in : `ft_perso.h`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a file `ft_perso.h` which compiles the following main :

```c
#include "ft_perso.h"

int main()
{
  t_perso   jack;

  jack.name = strdup("jack");
  jack.life = 100.0;
  jack.age = 42;
  jack.profession = SAVE_THE_WORLD;
  return (0);
}
```

Chapter III

ft_door

	Exercise 12
ft_door.h ft_door.c	
Turn-in directory : *ex12/*	
Files to turn in : `ft_door.h, ft_door.c`	
Allowed functions : `write`	
Notes : `n/a`	

- Create the file ft_door.h and fix the following ft_door.c file :

```
#include "ft_door.h"

ft_putstr(char *str)
{
        int i = 0;

        while (str[i])
                write(1, str, i)
}

ft_bool close_door(t_door *door)
{
        ft_putstr(Door closing...);
        state = CLOSE;
        return  (TRUE);
}

void is_door_open(t_door door)
{
        ft_putstr("Door is open ?");
        return (door->state = OPEN);
}

ft_bool is_door_close(t_door* door)
{
        ft_putstr("Door is close ?") ;
}
```

- Here's an example of main and output.

```
$> cat main.c
#include <stdlib.h>
#include "ft_door.h"

int main()
{
    t_door      door;

    open_door(&door);
    if (is_door_close(&door))
        open_door(&door);
    if (is_door_open(&door))
        close_door(&door);
    if (door.state == OPEN)
        close_door(&door);
    return (EXIT_SUCCESS);
}
$>./ft_door | cat -e
Door opening...$
Door is close ?$
Door is open ?$
Door closing...$
$>
```

Chapter III

ft_compact.c

	Exercise 13
ft_compact.c	
Turn-in directory : *ex*13/	
Files to turn in : `ft_compact.c`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a function `ft_compact` which takes a array of `char *` as argument and crushes all elements pointing to 0.
- Here's how it should be prototyped :

```
int   ft_compact(char **tab, int length);
```

- This function will return the array's new size.

`Not sure what this function does?  It also exists in Ruby.`