B1- Unix and C Lab Seminar

B-CPE-100

Day 04

Pointers

v1.6

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Day 04

Pointers

repository name:	CPool_Day04_$ACADEMICYEAR
repository rights:	ramassage-tek
language:	C
group size:	1

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- Don't push your **main** function into your delivery directory, we will be adding our own. Your files will be compiled adding our **main.c** and our **my_putchar.c** files.
- You are only allowed to use the **my_putchar** function to complete the following tasks, but don't push it into your delivery directory, and don't copy it in *any* of your delivered files.
- If one of your files prevents you from compiling with * **.c**, the Autograder will not be able to correct your work and you will receive a 0.

Create your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to `gitignore`.

Task 00

Unit Tests

It is highly recommended to test your functions as you develop them. It is common practice to create and use what is called **unit tests**.
From now on, we expect you to write unit tests for your functions (when possible). To do that, please follow the instructions on the document "How to write Unit Tests" from the intranet, available here.
Here's a few example of unit test for the my_strlen function:

```
#include <criterion/criterion.h>

Test(my_strlen, return_value_is_good)
{
        cr_assert_eq(my_strlen("toto"), 4);
        cr_assert_eq(my_strlen(""), 0);
        cr_assert_eq(my_strlen("Hello World!"), 12);
}
```

Task 01

my_swap

Write a function that swaps the content of two integers, whose addresses are given as a parameter. It must be prototyped as follows:

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

Delivery: CPool_Day04_$ACADEMICYEAR/my_swap.c

Task 02

my_putstr

Write a function that displays, one-by-one, the characters of a string.
The address of the string's first character will be found in the pointer passed as a parameter to the function, which must be prototyped as follows:

```
int my_putstr(char const *str);
```

Delivery: CPool_Day04_$ACADEMICYEAR/my_putstr.c

Task 03

my_strlen

Write a function that counts and returns the number of characters found in the string passed as parameter. It must be prototyped as follows:

```
int my_strlen(char const *str);
```

Delivery: CPool_Day04_$ACADEMICYEAR/my_strlen.c

Task 04

my_evil_str

The goal of this task is to swap each of the string's characters, two by two.
In other words, you will swap the first letter with the last one, the second with the second-to-last and so on.
The function should return a pointer to the first character of the reversed string:

```
char *my_evil_str(char *str);
```

Delivery: CPool_Day04_$ACADEMICYEAR/my_evil_str.c

For instance:

```
a => a
ab => ba
abc => cba
abcd => dcba
abcde => edcba
abcdef => fedcba
```

When testing your function you may encounter "Segmentation fault" errors. Either you're messing with the pointers in your function or the string given in parameter is read-only!

Easy way to have read/write string for testing purpose: *man 3 strdup*

Unit Test

WRITE THEM!

If you haven't covered all the possible case of the previous functions with unit test, it is really time to do so!

A good habit is to practice what is called Test Driven Development, you can read more about it here.

Task 05

my_getnbr

Write a function that returns a number, sent to the function as a string. It must be prototyped as follows:

```
int my_getnbr(char const *str);
```

Delivery: CPool_Day04_$ACADEMICYEAR/my_getnbr.c

Here are some tricky examples to be handled:

```
"+---+--++---+---+---+-42" => -42
"42a43" => 42
"1100000000000000000000042" => 0 (the number doesn't fit in an integer)
"-1000000000000000000000042" => 0 (for the same reason)
```

Task 06

my_sort_int_array

Write a function that sorts an integer array in ascending order, given a pointer to the first element of the array and its size.

```
void my_sort_int_tab(int *array, int size);
```

Delivery: CPool_Day04_$ACADEMICYEAR/my_sort_int_array.c